

Package: hvtiPlotR (via r-universe)

October 21, 2026

Type Package

Title HVTI 'ggplot2' Themes and Clinical Plot Functions for the
Cleveland Clinic Heart & Vascular Institute

Version 2.8.16

Date 2026-10-21

Maintainer John Ehrlinger <john.ehrlinger@gmail.com>

License GPL-3

Encoding UTF-8

URL <https://github.com/ehrlinger/hvtiPlotR>,
<https://ehrlinger.github.io/hvtiPlotR/>,
<https://ehrlinger.github.io/hvtiGraphics/>

BugReports <https://github.com/ehrlinger/hvtiPlotR/issues>

Description Themes, clinical plot constructors and methods for
creating publication quality 'ggplot2' graphics in R that
conform to the standards of Cardiovascular Outcomes Registries
and Research (CORR) within The Heart & Vascular Institute at
the Cleveland Clinic.

Depends R (>= 4.4.0)

Imports consort (>= 0.2.0), ggalluvial (>= 0.12.3), ggplot2 (>= 4.0.0), ggsankey, ggupset (>= 0.4.0), ggvenn, grid, patchwork (>= 1.3.1), rlang (>= 1.0.0), survival (>= 3.1-0), officer (>= 0.6.0), rvg (>= 0.2.5), ragg (>= 1.2.1), withr

Config/testthat/edition 3

Suggests knitr, ggforce, testthat (>= 3.0.0), devtools, haven, scales, tidyr, quarto, pkgload, here, xml2, cobalt (>= 5.0.0), gdtools (>= 0.5.0), ggiraph (>= 0.9.5), hvtiRpropensity

Remotes davidsjoberg/ggsankey, ehrlinger/hvtiRpropensity

VignetteBuilder quarto

SystemRequirements Quarto command line tool
(<<https://github.com/quarto-dev/quarto-cli>>)

Roxygen list(markdown = TRUE)

Config/roxygen2/version 8.1.0

Config/pak/sysreqs libcairo2-dev libfontconfig1-dev libfreetype6-dev libfribidi-dev libharfbuzz-dev libicu-dev libjpeg-dev libpng-dev libtiff-dev libwebp-dev libxml2-dev libssl-dev

Repository <https://ehrlinger.r-universe.dev>

Date/Publication 2026-10-21 18:10:58 UTC

RemoteUrl <https://github.com/ehrlinger/hvtiPlotR>

RemoteRef HEAD

RemoteSha 17cb76ba25918ad46187a88e3c3d214628bc7e08

Contents

hvtiPlotR-package	5
as.data.frame.hv_data	9
autoplot.hv_data	10
eda_classify_var	11
eda_select_vars	12
hv_add_plot	13
hv_alluvial	14
hv_atrisk	16
hv_atrisk_compose	17
hv_balance	18
hv_balance_height	21
hv_consort	21
hv_consort_exclude	23
hv_consort_patients	24
hv_consort_start	25
hv_consort_summary	26
hv_correlation_matrix	27
hv_eda	28
hv_eda_pages	30
hv_fit_location	32
hv_followup	34
hv_followup_panels	36
hv_ggsave_dims	38
hv_girafe	39
hv_hazard	40
hv_legend_inside	43
hv_longitudinal	44
hv_mirror_hist	46
hv_nnt	48
hv_nonparametric	49
hv_ordinal	52
hv_palette	53
hv_panel_box	54

hv_panel_size	56
hv_ph_location	57
hv_ppt_layout	58
hv_ppt_palette	59
hv_ppt_series	61
hv_rmst_contrast	63
hv_rmst_curves	65
hv_role_palette	67
hv_sankey	68
hv_spaghetti	70
hv_stacked	71
hv_survival	73
hv_survival_difference	75
hv_trends	76
hv_upset	78
hv_venn	79
hvtiPlotR-themes	80
is_hv_data	83
make_footnote	84
nonparametric	86
parametric	87
plot.hv_alluvial	89
plot.hv_balance	90
plot.hv_consort	92
plot.hv_correlation_matrix	92
plot.hv_eda	93
plot.hv_eda_pages	95
plot.hv_followup	96
plot.hv_followup_panels	97
plot.hv_hazard	98
plot.hv_longitudinal	99
plot.hv_mirror_hist	100
plot.hv_nnt	101
plot.hv_nonparametric	102
plot.hv_ordinal	103
plot.hv_rmst_contrast	104
plot.hv_rmst_curves	105
plot.hv_sankey	107
plot.hv_spaghetti	108
plot.hv_stacked	110
plot.hv_survival	111
plot.hv_survival_difference	112
plot.hv_trends	113
plot.hv_upset	115
plot.hv_venn	117
print.hv_alluvial	119
print.hv_balance	119
print.hv_consort	120

print.hv_consort_tracker	120
print.hv_eda	121
print.hv_eda_pages	121
print.hv_followup	122
print.hv_followup_panels	122
print.hv_hazard	123
print.hv_longitudinal	123
print.hv_mirror_hist	124
print.hv_nnt	124
print.hv_nonparametric	125
print.hv_ordinal	125
print.hv_rmst_contrast	126
print.hv_rmst_curves	126
print.hv_sankey	127
print.hv_spaghetti	127
print.hv_stacked	128
print.hv_survival	128
print.hv_survival_difference	129
print.hv_trends	129
print.hv_upset	130
print.hv_venn	130
sample_alluvial_data	131
sample_cluster_sankey_data	131
sample_consort_data	133
sample_correlation_data	133
sample_covariate_balance_data	134
sample_eda_data	135
sample_goodness_followup_data	136
sample_hazard_cohort	138
sample_hazard_data	140
sample_hazard_empirical	142
sample_life_table	143
sample_longitudinal_counts_data	144
sample_mirror_histogram_data	145
sample_nnt_data	147
sample_nonparametric_curve_data	148
sample_nonparametric_curve_points	149
sample_nonparametric_ordinal_data	151
sample_nonparametric_ordinal_points	152
sample_spaghetti_data	153
sample_stacked_histogram_data	154
sample_survival_data	155
sample_survival_difference_data	156
sample_trends_data	158
sample_upset_data	159
save_manuscript	159
save_ppt	161
scale_fill_hv	167

summary.hv_data	168
---------------------------	-----

Index	170
--------------	------------

hvtiPlotR-package	<i>hvtiPlotR: Publication-Quality Graphics for Clinical Manuscripts and Slides</i>
-------------------	--

Description

hvtiPlotR is an R port of the `plot.sas` macro suite used by the Cardiovascular Outcomes, Registries and Research (CORR) statistics group within the Heart, Vascular and Thoracic Institute at the Cleveland Clinic. We use it to produce graphics that meet HVTI manuscript, poster, and presentation standards, built on `ggplot2` and the `officer` package.

Worked, rendered examples for every constructor and theme live in the companion **HVTI ggplot graphics recipes** book, <https://ehrlinger.github.io/hvtiGraphics/>.

Details

Two-step workflow:

Every plot constructor follows the same pattern:

1. Call `hv_*`() to validate and prepare data. It returns an `hv_data` S3 object that also stores diagnostics (group counts, SMDs, etc.) under `$tables`.
2. Call `plot()` on the object to obtain a bare `ggplot2::ggplot` you can decorate with scales, labels, annotations, and themes without restriction.

Calling `print()` on the object gives a concise console summary.

```
library(ggplot2)
library(hvtiPlotR)
```

```
dta <- sample_trends_data(n = 600, seed = 42)
tr <- hv_trends(dta)
```

```
plot(tr) +
  scale_color_manual(values = hv_palette(), name = "Group") +
  labs(x = "Year", y = "Outcome") +
  theme_hv_manuscript()
```

Every constructor ships with a `sample_*`() companion that generates realistic synthetic data sized and shaped like the corresponding SAS dataset export, so examples and unit tests run without PHI.

Themes:

Four `theme_hv_*`() functions follow `ggplot2`'s `theme_bw()` contract: pass `base_size/base_family` to control typography, then forward any extra named theme element through ... (e.g. `theme_hv_manuscript(legend.position = "right")`) or chain a `+ theme(...)` call.

- `theme_hv_manuscript()`: Clean white-background theme for journal figures.
- `theme_hv_poster()`: Medium-font theme with visible axis lines for conference posters.

- `theme_hv_ppt_dark()`: Large-font theme with a black panel for blue-gradient or dark PowerPoint templates.
- `theme_hv_ppt_light()`: Large-font theme with a transparent panel (appears white on white-background PowerPoint templates). Add `+ theme(panel.background = element_rect(fill = "white"))` if you specifically need an opaque white panel on a non-white slide background.
- `theme_hv_ppt_slate()`: Large-font theme with a white panel and white axes for the slate template (`hv_ppt_template_slate.pptx`).
- `hv_ppt_layout()`: The layout name for a slide role; the same six layouts exist in every bundled template.

All three PPT themes hide the legend by default, use inside-facing axis ticks, and scale axis-text / axis-title margins from `base_size` via `ggplot2`'s `half_line` convention.

Saving figures:

- `save_manuscript()`: Save a ggplot at the house manuscript size (6 x 4 in) in one call; the manuscript counterpart of `save_ppt()`. Pair with `theme_hv_manuscript()` for 12 pt type.
- `save_ppt()`: Insert one or more ggplot objects into a PowerPoint file as editable DrawingML vector graphics via `officer` + `rvg`.
- `make_footnote()` / `makeFootnote()`: Add a draft footnote during analysis; omit for publication-ready output.

Fixed-panel geometry (v2.0.0):

When plots in a set have different axis-label widths (e.g. y-axis ranges from "1.0" to "4567.2"), the usual `ggsave()` and `officer::ph_location()` calls let the panel content area drift, which is visually jarring on PPT decks where the black or white panel box should appear constant across slides. Two helpers solve this by making the panel size and slide position the *target*:

- `hv_ggsave_dims()`: Given a target panel width and height, computes the `ggsave()` width/height that preserve that panel size. Returns a named list to splat into `ggsave()` via `do.call(ggplot2::ggsave, c(list(filename = ..., plot = p), dims))`.
- `hv_ph_location()`: Given a target panel rectangle on the slide, computes the `officer::ph_location()` width/height/left/top that anchor the panel to that rectangle regardless of axis-label width. Warns if chrome overflows the left or top slide edge.
- `save_ppt()` accepts `panel_box = list(width = ..., height = ..., left = ..., top = ...)` to apply `hv_ph_location()` per slide automatically.

Plot constructors:

Propensity Score & Matching:

- `hv_mirror_hist()`: Prepare propensity-score distributions for a mirrored histogram comparing binary-matched or IPTW-weighted cohorts. Ports the `tp.lp.mirror-histogram_*` and `tp.lp.mirror_histo_before_after_wt` SAS scripts.
- `hv_balance()`: Prepare standardized mean difference data for a covariate balance dot-plot before and after propensity-score matching or weighting. Ports `tp.lp.propen.cov_balance.R`.
- `hv_stacked()`: Prepare grouped count or proportion data for a stacked or filled histogram.

Survival & Hazard:

- `hv_survival()`: Fit and prepare Kaplan-Meier or Nelson-Aalen survival curves, risk tables, and report tables. Ports the SAS `%kaplan` and `%nelson` macros from `tp.ac.dead.sas`.

- `hv_hazard()`: Prepare parametric hazard / survival curves from pre-fitted model output, with optional KM empirical overlay and population life-table reference. Ports the `tp.hp.*` template family.
- `hv_survival_difference()`: Prepare mean survival difference (life gained) over time between two treatment arms, with confidence ribbons. Ports `tp.hp.dead.life-gained.sas`.
- `hv_nnt()`: Prepare number-needed-to-treat over time derived from the survival difference. Ports `tp.hp.numtreat.survdiff.matched.sas`.

Nonparametric Temporal Curves:

- `hv_nonparametric()`: Prepare pre-computed two-phase nonparametric temporal trend curves for binary or continuous outcomes with optional 68%/95% CI ribbon and binned data summary points. Ports the `tp.np.*` template family.
- `hv_ordinal()`: Prepare grade-specific probability curves from cumulative proportional-odds models (e.g. TR / AR grade). Ports `tp.np.tr.ivecho.*` and `tp.np.po_ar.u_multi.ordinal.sas`.

Study Design & Goodness of Follow-Up:

- `hv_followup()`: Prepare per-patient follow-up data for a goodness-of-follow-up scatter plot (actual vs. potential follow-up per operation year). Use `plot(x, type = "followup")` for the death panel and `plot(x, type = "event")` for the non-fatal event panel. Ports `tp.dp.gfup.R`.

Longitudinal & Repeated Measures:

- `hv_trends()`: Prepare patient-level data for annual trend lines with optional confidence ribbons for multiple groups. Ports the `tp.lp.trends.*`, `tp.rp.trends.*`, and `tp.dp.trends.R` families.
- `hv_spaghetti()`: Prepare repeated-measures data for individual patient trajectory plots, optionally stratified by group. Ports `tp.dp.spaghetti.echo.R`.
- `hv_longitudinal()`: Prepare pre-aggregated counts for a grouped bar chart or text table of patient and measurement counts at discrete follow-up windows. Use `plot(x, type = "plot")` for the bar chart and `plot(x, type = "table")` for the numeric table panel.

Flow Diagrams:

- `hv_alluvial()`: Prepare patient-flow data for an alluvial (Sankey) diagram showing transitions between states across time points. Ports `tp.dp.female_bicus_preAR_sankey.R`.
- `hv_sankey()`: Prepare cluster-assignment data for a stability Sankey showing how patients redistribute across K values in a PAM analysis.

Exploratory Data Analysis:

- `hv_eda()`: Prepare a single variable for EDA plotting: bar chart (categorical) or scatter + LOESS (continuous). Ports `tp.dp.EDA_barplots_scatterplots.R`.
- `eda_classify_var()`: Classify a vector as "Cont", "Cat_Num", or "Cat_Char" using the UniqueLimit heuristic from `Barplot_Scatterplot_Function.R`.
- `eda_select_vars()`: Subset and reorder a data frame by a character vector or space-separated column-name string. Replaces `Order_Variables()`.
- `hv_upset()`: Prepare binary indicator data for an UpSet intersection plot. Ports `tp.complexUpset.R`.

Legacy single-call plot API:

These functions predate the `constructor-and-plot()` split and return a `ggplot` directly. They are retained for backward compatibility with scripts written before the `hv_*` redesign. New code should prefer the constructor form.

- `hazard_plot()`: Single-call parametric hazard/survival plot, combine-step equivalent of `hv_hazard() + plot()`.
- `survival_difference_plot()`: Single-call mean-survival-difference plot, equivalent of `hv_survival_difference() + plot()`.
- `nnt_plot()`: Single-call NNT plot, equivalent of `hv_nnt() + plot()`.

Data class and introspection:

All `hv_*`() constructors return objects that inherit from `hv_data` with a concept-specific subclass (e.g. `hv_survival`, `hv_trends`).

- `is_hv_data()`: Predicate for the shared base class.
- `inherits(x, "hv_trends")`: Check for a specific constructor's output.
- `x$data`: The canonical long-format data frame used by `plot(x)`.
- `x$meta`: Named list of structural metadata (column names, group levels, summary functions).
- `x$tables`: Concept-specific diagnostic or report tables (SMDs, risk tables, group counts, etc.).

Sample-data generators:

- `sample_mirror_histogram_data()`: Propensity scores via a logistic model with greedy 1:1 caliper matching and optional IPTW weights.
- `sample_covariate_balance_data()`: Standardized mean differences before and after propensity matching.
- `sample_stacked_histogram_data()`: Year-by-category count data.
- `sample_survival_data()`: Exponential survival times with administrative censoring and optional treatment strata.
- `sample_hazard_data()`: Weibull parametric survival predictions on a fine time grid, matching the predict dataset from `%hazpred`.
- `sample_hazard_empirical()`: Binned Kaplan-Meier empirical points matching the `plout` / `acpdms` dataset.
- `sample_life_table()`: Age-group-specific Gompertz survivorship curves matching US population life-table SAS overlays.
- `sample_survival_difference_data()`: Mean survival difference between two parametric arms with confidence bands.
- `sample_nnt_data()`: Number-needed-to-treat derived from survival difference data.
- `sample_nonparametric_curve_data()`: Two-phase nonparametric curve predictions (binary or continuous outcome) on a fine time grid.
- `sample_nonparametric_curve_points()`: Binned patient-level data summary points matching the SAS means dataset.
- `sample_nonparametric_ordinal_data()`: Grade-specific cumulative proportional-odds probability curves.
- `sample_nonparametric_ordinal_points()`: Binned ordinal data summary points per grade level.
- `sample_goodness_followup_data()`: Operative cohort with operation dates, follow-up times, death, and non-fatal events.
- `sample_trends_data()`: Multi-group annual trend data with confidence ribbons.
- `sample_spaghetti_data()`: Patient-level longitudinal measurements over time.

- `sample_longitudinal_counts_data()`: Pre-aggregated patient and measurement counts at discrete follow-up windows.
- `sample_alluvial_data()`: Patient flow between states across multiple time points in wide format.
- `sample_cluster_sankey_data()`: Cluster assignments across K values for a PAM stability analysis.
- `sample_eda_data()`: Mixed-type cardiac-surgery registry simulation (binary, ordinal, character-categorical, and continuous variables).
- `sample_upset_data()`: Binary indicator columns for UpSet intersection analysis.

Scope and versioning:

hvtiPlotR targets internal HVTI / CORR use only; it will not be submitted to CRAN. GitHub-only dependencies are declared in Remotes: in DESCRIPTION and install cleanly. Install via:

```
remotes::install_github("ehrlinger/hvtiPlotR")
```

Releases follow straight semantic versioning (2.0.0, 2.0.1, 2.1.0, 3.0.0 ...). The main branch tracks the most recent release; engineering collects changes staged for the next release. No .9xxx pre-release suffix on development builds. Pin against a specific release tag for reproducibility, e.g. `remotes::install_github("ehrlinger/hvtiPlotR@v2.0.0")`.

Vignettes:

- `vignette("hvtiPlotR")`: Introductory tutorial paralleling the SAS `plot.sas` macro, with side-by-side SAS and R examples.
- `vignette("plot-functions")`: Worked examples for every `hv_*`() constructor with the three-step build/bare-plot/decorate pattern.
- `vignette("plot-decorators")`: Guide to scales, labels, annotations, coord crops, themes, and `ggsave()` / `save_ppt()` patterns, including the `panel_box` fixed-panel workflow.
- `vignette("sas-migration-guide")`: SAS `plot.sas` template name lookup mapping to the corresponding `hv_*`() constructor and a worked R example.

References

Wickham, H. *ggplot2: Elegant Graphics for Data Analysis*. Springer, 2009.

Gohel, D. *officer: Manipulation of Microsoft Word and PowerPoint Documents*. R package. <https://davidgohel.github.io/officer/>

as.data.frame.hv_data *Extract the underlying data frame from an hv_data object*

Description

Returns the `$data` slot, the tidy data frame each `hv_data` subclass carries for `ggplot2` consumption. Lets callers use the standard `base::as.data.frame()` / `base::data.frame()` coercion in tidyverse pipelines instead of reaching for the `$data` accessor.

Usage

```
## S3 method for class 'hv_data'
as.data.frame(x, row.names = NULL, optional = FALSE, ...)
```

Arguments

x	An hv_data object.
row.names	Ignored; present for base-method-signature consistency.
optional	Ignored; present for base-method-signature consistency.
...	Ignored.

Value

The data.frame stored in x\$data.

autoplot.hv_data	<i>Produce a ggplot from an hv_data object (autoplot generic)</i>
------------------	---

Description

Re-exports ggplot2's `ggplot2::autoplot()` for hv_data objects so callers who prefer the ggplot2-ecosystem verb (broom / ggfortify / ggsurvfit all use it) can write `autoplot(km)` interchangeably with `plot(km)`.

Usage

```
## S3 method for class 'hv_data'
autoplot(object, ...)
```

Arguments

object	An hv_data object.
...	Forwarded to the subclass's plot() method.

Details

Dispatches to the registered `plot.<subclass>()` method, so any args that work with `plot()` work with `autoplot()`.

Value

A ggplot object (or a patchwork composite when the underlying plot() method returns one, e.g. for hv_upset).

eda_classify_var	<i>Classify a Variable as Continuous or Categorical</i>
------------------	---

Description

Replicates the type-detection logic from `Barplot_Scatterplot_Function.R`: a numeric column is treated as categorical when all non-missing values are non-negative whole numbers with no more than `unique_limit` distinct values.

Usage

```
eda_classify_var(
  x,
  unique_limit = 6L,
  unique_bound = 100,
  var_name = NULL,
  type_overrides = NULL
)
```

Arguments

<code>x</code>	A vector (one column of a data frame).
<code>unique_limit</code>	Integer threshold. Numeric columns with more distinct values than this are classified as "Cont". Default 6.
<code>unique_bound</code>	Integer threshold. Numeric columns that contain any values greater than this are classified as "Cont", regardless of number of distinct values. Default 100
<code>var_name</code>	A string containing the name of the variable/column. Default NULL
<code>type_overrides</code>	A named vector of variables and variable types ("Cont", "Cat_Num", or "Cat_Char") to manually set their classifications, e.g., <code>c(severity = "Cat_Num")</code> . Default NULL

Value

A length-1 character: "Cont", "Cat_Num", or "Cat_Char".

See Also

[hv_eda\(\)](#), [eda_select_vars\(\)](#)

Examples

```
eda_classify_var(c(0, 1, 1, 0, NA))      # "Cat_Num"
eda_classify_var(c(1, 2, 3, 4))          # "Cat_Num"
eda_classify_var(rnorm(50))              # "Cont"
eda_classify_var(c("A", "B", "A"))       # "Cat_Char"
```

eda_select_vars	Select and Reorder Variables from a Data Frame
-----------------	--

Description

Returns the subset of data containing only the columns named in `vars`, in the order given. Replaces the `Order_Variables()` helper and the `Mod_Data <- dta[, Order_Var]` pattern from `tp.dp.EDA_barplots_scatterplots`.

Usage

```
eda_select_vars(data, vars)
```

Arguments

<code>data</code>	A data frame.
<code>vars</code>	A character vector of column names, or a single string of space-separated names, e.g. "age ht wt bmi".

Details

`vars` may be supplied as a character vector or as a single space-separated string (matching the `Var_CatList` / `Var_ContList` style from the template).

Value

A data frame containing only the requested columns in the requested order.

See Also

[hv_eda\(\)](#)

Examples

```
dta <- sample_eda_data()

# Vector form
sub <- eda_select_vars(dta, c("male", "cabg", "nyha"))
names(sub)

# Space-separated string (matches template Var_CatList style)
sub2 <- eda_select_vars(dta, "male cabg nyha valve_morph")
names(sub2)
```

hv_add_plot	<i>Place one plot on the current slide, fitted around the standard panel</i>
-------------	--

Description

Adds plot to the current slide of an officer presentation with its panel anchored to panel_box, so every plot slide of a slide format puts its panel in the same place. With `fit = TRUE` the plot's text sizes and text margins are reduced together by `hv_fit_location()` until axis text, axis titles, facet strips and plot titles lie on the slide and below its title.

Usage

```
hv_add_plot(
  doc,
  plot,
  panel_box = hv_panel_box(format = .hv_doc_format(doc)),
  fit = TRUE,
  png_fallback = TRUE,
  dpi = 300,
  ...
)
```

Arguments

doc	An rpptx object from <code>officer::read_pptx()</code> whose current slide receives the plot.
plot	A ggplot or patchwork object.
panel_box	Named list <code>list(width, height, left, top)</code> , in inches. Defaults to <code>hv_panel_box()</code> for the slide format of doc, read from its slide size: 16:9 (every bundled template) or 4:3. A deck of any other size needs an explicit panel_box.
fit	TRUE (the default) to reduce the plot's text until the figure fits on the slide. FALSE places the plot at its own sizes and warns if it runs off the slide or into the title.
png_fallback	TRUE (the default) to place a PNG when the vector insert fails. FALSE lets the error through.
dpi	Resolution of the fallback PNG. Default 300.
...	Further arguments to <code>hv_fit_location()</code> : <code>title_bottom</code> , <code>edge</code> , <code>min_scale</code> , <code>step</code> , <code>align</code> , <code>margin</code> . The slide size is read from doc. <code>align = "fill"</code> widens the panel so the figure spans the slide, centered, instead of anchoring it to panel_box.

Details

The plot goes in as an editable DrawingML vector graphic via `rvg::dml()`. A dense point cloud can overflow libxml2 and make that insert fail; with `png_fallback = TRUE` the plot is then rendered to a transparent PNG at exactly the fitted location's size and placed at the same location, so the panel still lands on panel_box.

Add the slide and its title first, with `officer::add_slide()` and `officer::ph_with()`; `save_ppt()` does all of this for a list of plots.

Value

doc, with the plot added, carrying an attribute "hv_fit": a list scale (the text scale used), fits (whether the figure lies inside the slide limits), png (whether the PNG fallback was used), location (the placement, in inches) and panel_box (the panel rectangle used). A message reports reduced text; a warning reports a figure that still does not fit.

See Also

`save_ppt()`, `hv_fit_location()`, `hv_panel_box()`.

Examples

```
p <- ggplot2::ggplot(mtcars, ggplot2::aes(hp, mpg)) +
  ggplot2::geom_point() +
  ggplot2::labs(x = "Horsepower", y = "Miles per gallon") +
  theme_hv_ppt_dark()

template <- system.file("extdata", "hv_ppt_template.pptx",
  package = "hvtiPlotR")
doc <- officer::read_pptx(template)
doc <- officer::add_slide(doc, layout = hv_ppt_layout("figure"))
doc <- officer::ph_with(doc, "Fuel economy",
  location = officer::ph_location_type("title"))
doc <- hv_add_plot(doc, p)
attr(doc, "hv_fit")$scale
print(doc, target = file.path(tempdir(), "fitted.pptx"))
```

hv_alluvial

Prepare alluvial / Sankey diagram data for plotting

Description

Validates a wide alluvial-format data frame and returns an `hv_alluvial` object. Call `plot.hv_alluvial` on the result to obtain a bare `ggplot2` alluvial diagram that you can decorate with color scales, axis labels, and `theme_hv_manuscript`.

Usage

```
hv_alluvial(data, axes, y_col = "freq", fill_col = NULL, axis_labels = NULL)
```

Arguments

data	A data frame in wide alluvial format: one row per axis-value combination, a numeric weight column, and one column per axis.
axes	Character vector of column names to use as axes, in left-to-right display order. Minimum two columns.
y_col	Name of the numeric weight column (counts or proportions). Default "freq".
fill_col	Name of the column to map to the flow fill and color aesthetics, or NULL for a single fill. Default NULL.
axis_labels	Character vector of axis labels for the x-axis, the same length as axes. Defaults to axes (column names).

Value

An object of class `c("hv_alluvial", "hv_data")`:

`$data` The validated input data frame.

`$meta` Named list: axes, y_col, fill_col, axis_labels, n_axes, n_obs.

`$tables` Empty list.

See Also

Worked recipe with rendered output: <https://ehrlinger.github.io/hvtiGraphics/sankey.html>.

`plot.hv_alluvial`, `sample_alluvial_data`

Examples

```
dta <- sample_alluvial_data(n = 300, seed = 42)
axes <- c("pre_ar", "procedure", "post_ar")

# 1. Build data object
al <- hv_alluvial(dta, axes = axes, y_col = "freq", fill_col = "pre_ar")
al # prints axes and observation count

# 2. Bare plot -- undecorated ggplot returned by plot.hv_alluvial
p <- plot(al)

# 3. Decorate: fill/color palettes, axis labels, theme
p +
  ggplot2::scale_fill_manual(
    values = c(None      = "steelblue",
               Mild      = "goldenrod",
               Moderate  = "darkorange",
               Severe    = "firebrick"),
    name = "Pre-op AR"
  ) +
  ggplot2::scale_color_manual(
    values = c(None      = "steelblue",
               Mild      = "goldenrod",
```

```

        Moderate = "darkorange",
        Severe   = "firebrick"),
    guide = "none"
  ) +
  ggplot2::labs(y = "Patients (n)",
               title = "AV Regurgitation: Pre- to Post-operative") +
  theme_hv_poster()

```

hv_atrisk

Numbers-at-risk table panel

Description

Renders a numbers-at-risk table as a bare **ggplot2** panel, ready to stack under a survival curve with [hv_atrisk_compose](#).

Usage

```

hv_atrisk(
  x,
  time = NULL,
  status = NULL,
  group = NULL,
  report_times = NULL,
  size = NULL,
  strata_labels = NULL,
  ...
)

```

Arguments

x	One of: an <code>hv_data</code> object that carries <code>\$tables\$risk</code> (e.g. from hv_survival); a precomputed risk data frame with <code>strata</code> , <code>report_time</code> (or <code>time</code>), and <code>n.risk</code> (or <code>n</code>) columns; or a subject-level data frame supplied together with the <code>time</code> (and optional <code>status/group</code>) column names, from which counts are computed.
time, status, group	Column names in <code>x</code> when <code>x</code> is a subject-level data frame. <code>time</code> triggers the raw-data path; <code>status</code> , when named, leaves out rows whose status is missing (with a warning), as hv_survival does, so the two tables agree; <code>group</code> splits the table into strata. Pass <code>group</code> as a factor to control the stratum row order (its levels set the order); a character column orders rows alphabetically. Default <code>NULL</code> .
report_times	Numeric time points for the columns. <code>NULL</code> (default) uses the table's own points or, on the raw-data path, an even spread derived from the observed time range. On the object and precomputed-table paths a non- <code>NULL</code> value <i>selects</i> which of the table's existing times to show; the counts are not recomputed, and any requested time not in the table is ignored with a warning. To use arbitrary times

for a Kaplan-Meier object, rebuild it with `hv_survival(..., report_times = )` or pass the subject-level data.

`size` Text size for the counts. Default NULL (3.5).

`strata_labels` Optional named character vector remapping stratum row labels. Default NULL.

`...` Ignored; present for S3 consistency.

Value

A bare `ggplot` object: one text label per stratum and report time, strata as y rows (first stratum on top), time on a continuous x with axis text blanked (the curve carries the axis).

See Also

Worked recipe with rendered output: <https://ehrlinger.github.io/hvtiGraphics/survival.html>.

[hv_atrisk_compose](#), [hv_survival](#)

Examples

```
km <- hv_survival(sample_survival_data(n = 200, seed = 1))
hv_atrisk(km)
```

hv_atrisk_compose	<i>Stack a survival curve over a numbers-at-risk table</i>
-------------------	--

Description

Composes a curve plot over a `hv_atrisk` panel, aligning the table's x-range to the curve's so the counts sit under the matching times, and stacks them with **patchwork**.

Usage

```
hv_atrisk_compose(curve, table, heights = c(3, 1))
```

Arguments

`curve` A **ggplot2** survival curve, e.g. `plot(hv_survival(...))`.

`table` A **ggplot2** numbers-at-risk panel from `hv_atrisk`.

`heights` Numeric length-2 curve:table height ratio. Default `c(3, 1)`.

Value

A **patchwork** object (the curve above, the table below). Decorate both panels with `patchwork's &`, e.g. `& theme_hv_manuscript()`.

See Also[hv_atrisk](#)**Examples**

```
km <- hv_survival(sample_survival_data(n = 200, seed = 1))
hv_atrisk_compose(plot(km), hv_atrisk(km))
```

hv_balance

*Prepare covariate balance data for plotting***Description**

Validates and orders a long-format standardized-mean-difference data frame for a covariate balance plot, and returns an `hv_balance` object. Call [plot.hv_balance](#) on the result to obtain a bare `ggplot2` object that you can decorate with color, shape, axis scales, and [theme_hv_manuscript](#).

Usage

```
hv_balance(
  data,
  variable_col = "variable",
  group_col = "group",
  std_diff_col = "std_diff",
  var_levels = NULL,
  threshold = 10,
  stat_col = NULL,
  pair_col = NULL,
  stats = "smd",
  vr_threshold = c(0.5, 2),
  ks_threshold = NULL,
  group_labels = c("Before", "After")
)
```

Arguments

<code>data</code>	A data frame in long format with one row per covariate $\times$ group combination. Wide-format data must be reshaped first (e.g. with <code>tidyr::pivot_longer()</code>). It may instead be a <code>cobalt::bal.tab()</code> result (built with <code>un = TRUE</code> to draw the Before group), an <code>hvtiRpropensity</code> <code>ps_*</code> object, or a list of those, one per imputation; see Details.
<code>variable_col</code>	Name of the column containing covariate labels. Default "variable".
<code>group_col</code>	Name of the column identifying the comparison group (e.g. "Before match" / "After match"). Default "group".
<code>std_diff_col</code>	Name of the numeric column holding standardized mean difference values. Default "std_diff".

var_levels	Character vector controlling the display order of covariates on the y-axis. The first element appears at the bottom. Defaults to the order of first appearance in <code>data[[variable_col]]</code> .
threshold	Numeric; absolute SMD value at which dotted reference lines are drawn ($\pm$ threshold). Default 10.
stat_col	Optional name of a column holding the statistic each row reports: "smd", "variance_ratio" or "ks". When NULL (default) every row is treated as "smd".
pair_col	Optional name of a column identifying the comparison (e.g. "B vs. A") when more than two groups were compared. Default NULL (a single comparison).
stats	Character vector of statistics to keep, in panel order; any of "smd", "variance_ratio", "ks". Default "smd". An error is raised if a requested statistic has no rows.
vr_threshold	One or two positive numbers; reference lines for the variance-ratio panel. Default <code>c(0.5, 2)</code> .
ks_threshold	NULL (no line, default) or a single number in <code>[0, 1]</code> ; reference line for the KS panel.
group_labels	Length-2 character vector naming the before and after groups. Default <code>c("Before", "After")</code> .

Details

Instead of a long data frame, data may be a `cobalt::bal.tab()` result, a multi-category `bal.tab` (one facet row per pair), an `hvtiRpropensity` `ps_match()` / `ps_weight()` object, or a list of any one of those, one per imputation. In those cases the SMD is multiplied by 100 and the `*_col` arguments are ignored.

A `bal.tab()` result must be built with `binary = "std"` and `continuous = "std"` (the default), so every difference is standardized, and with `un = TRUE` to draw the Before group: `cobalt` leaves the unadjusted columns empty otherwise, and `hv_balance()` then says so in a message and draws the After group only. Ask for the variance ratio and KS statistic with `stats = c("m", "v", "ks")`; the variance ratio is undefined for binary covariates, so those rows are left out of its panel. A distance or prop. score row (Type "Distance") is drawn as a covariate row, as in `cobalt::love.plot()`. `ps_*` objects carry standardized differences only.

A list is pooled across imputations: SMD and KS statistic are averaged, the variance ratio takes the geometric mean, and an NA in any imputation makes the pooled value NA (counted in `$meta$n_missing`). Every imputation must have the same rows, with one allowance: a covariate `cobalt` types as binary in some imputations (because it takes two values there) has no variance ratio in those, so its pooled variance ratio is NA.

Value

An object of class `c("hv_balance", "hv_data")`: a list with three elements:

`$data` The input data frame with a `cb_index` column added for y-axis positioning, a `stat` factor (levels = stats) and a `pair` column (NA when there are no comparisons, otherwise a factor in order of first appearance). If data already has its own `stat` or `pair` column it is left untouched and these values go in `.cb_stat` / `.cb_pair`.

`$meta` Named list: `variable_col`, `group_col`, `std_diff_col`, `var_levels`, `threshold`, `n_vars`, `n_groups`, `stats`, `pairs`, `stat_var` and `pair_var` (the names of the columns holding the statistic and the comparison), `n_imputations`, `vr_threshold`, `ks_threshold`, `suggested_height`.

`$tables` Empty list (no accessory tables).

See Also

Worked recipe with rendered output: <https://ehrlinger.github.io/hvtiGraphics/balance.html>.

`plot.hv_balance`, `sample_covariate_balance_data`

Examples

```
library(ggplot2)
dta <- sample_covariate_balance_data()

# 1. Build data object
cb <- hv_balance(dta)
cb # prints variable count, group count, threshold

# 2. Bare plot -- undecorated ggplot returned by plot.hv_balance
p <- plot(cb)

# 3. Decorate: color/shape palettes, x-axis range, labels, theme
p +
  scale_color_manual(
    values = c("Before match" = "red4", "After match" = "blue3"),
    name = NULL
  ) +
  scale_shape_manual(
    values = c("Before match" = 17L, "After match" = 15L),
    name = NULL
  ) +
  scale_x_continuous(limits = c(-45, 35), breaks = seq(-40, 30, 10)) +
  labs(x = "Standardized difference (%)", y = "") +
  theme_hv_poster()

# 4. From a cobalt::bal.tab() result: SMD, variance ratio and KS panels
if (requireNamespace("cobalt", quietly = TRUE)) {
  lal <- cobalt::lalone
  ps <- fitted(glm(treat ~ age + educ + race + married + nodegree + re74 + re75,
    data = lal, family = binomial()))
  bt <- cobalt::bal.tab(
    treat ~ age + educ + race + married + nodegree + re74 + re75,
    data = lal, weights = ifelse(lal$treat == 1, 1 / ps, 1 / (1 - ps)),
    binary = "std", s.d.denom = "pooled", stats = c("m", "v", "ks"), un = TRUE
  )
  plot(hv_balance(bt, stats = c("smd", "variance_ratio", "ks")))
}
```

hv_balance_height	<i>Suggested figure height for a covariate balance plot</i>
-------------------	---

Description

Covariate labels overprint when a balance plot is drawn too short. This returns a height that leaves about 0.17 in per covariate row at 12 pt, plus room for the x axis, its title and a legend, scaled by base_size.

Usage

```
hv_balance_height(n_rows, base_size = 12, n_pairs = 1)
```

Arguments

n_rows	Number of covariate rows (length(var_levels)).
base_size	Theme base font size in points. Default 12.
n_pairs	Number of pairwise-comparison facet rows. Default 1.

Value

Height in inches, a numeric scalar.

See Also

[hv_balance\(\)](#), whose `$meta$suggested_height` holds this value at `base_size = 12`.

Examples

```
hv_balance_height(38)           # about 8 in
hv_balance_height(38, base_size = 16)
hv_balance_height(12, n_pairs = 3) # three-group design
```

hv_consort	<i>Build a CONSORT flow diagram from a tracker</i>
------------	--

Description

Reads the stage metadata stored in an `hv_consort_tracker`, auto-derives the orders and `side_box` arguments for `consort::consort_plot()`, and returns an `hv_consort` object wrapping the grid diagram.

Usage

```

hv_consort(
  tracker,
  side_box = "all",
  cex = 0.9,
  width = NULL,
  height = NULL,
  ...
)

```

Arguments

tracker	An hv_consort_tracker with at least two stages (call hv_consort_exclude() at least once after hv_consort_start()).
side_box	Character vector of exclusion-reason column names to display as side boxes, or "all" (default) to include every exclusion column.
cex	Numeric; text size scaling passed to <code>consort::consort_plot()</code> . Default 0.9.
width	Diagram width in inches. Defaults to 7.
height	Diagram height in inches. Defaults to $2 + n_stages * 1.2$, where <code>n_stages</code> is the number of stages in the tracker.
...	Additional arguments forwarded to <code>consort::consort_plot()</code> .

Value

An hv_consort object, a list with:

`$plot` The grid object returned by `consort::consort_plot()`.

`$meta` Named list: `n_stages`, `width`, `height`, `orders`, `side_box`.

`$tracker` The original hv_consort_tracker.

See Also

Worked recipe with rendered output: <https://ehrlinger.github.io/hvtiGraphics/consort.html>.

[hv_consort_start\(\)](#), [hv_consort_exclude\(\)](#), [plot.hv_consort\(\)](#)

Examples

```

cohort <- data.frame(
  mrn = paste0("P", 1:100),
  age = sample(15:80, 100, TRUE),
  echo = sample(c(TRUE, FALSE), 100, TRUE, prob = c(0.9, 0.1))
)
tracker <- hv_consort_start(cohort, patient_id = mrn) |>
  hv_consort_exclude(label = "Eligible", col = "excl_screen",
    age < 18 ~ "Age < 18") |>
  hv_consort_exclude(label = "Analyzed", col = "excl_eligible",

```

```

                                !echo ~ "Missing echocardiogram")
fig <- hv_consort(tracker)
plot(fig)

```

hv_consort_exclude *Add an exclusion stage to a CONSORT tracker*

Description

Evaluates formula-based exclusion rules against the currently-active patient population and appends two new columns to the tracker's data frame: a character column (col) recording the first-matching exclusion reason for each patient, and a boolean column (pass_col) marking the survivors. Patients already excluded by a prior stage are automatically gated out.

Usage

```

hv_consort_exclude(
  tracker,
  label,
  col,
  ...,
  excl_label = "Excluded",
  pass_col = NULL
)

```

Arguments

tracker	An hv_consort_tracker from hv_consort_start() .
label	Character label for the survivor box after this exclusion (e.g. "Eligible", "Analyzed").
col	Column name to store exclusion reasons (character). This column will contain a reason string for excluded patients and NA for survivors. Required; there is no default.
...	Two-sided formulas of the form <condition> ~ "<reason string>". Conditions are evaluated with data masking against the tracker's data frame. The first matching formula assigns the reason; subsequent formulas are not evaluated for already- excluded patients.
excl_label	Character label for the side-box showing exclusion breakdown. Default "Excluded".
pass_col	Column name for the survivor boolean column. Defaults to ct_snakify(label) (e.g. "eligible" when label = "Eligible").

Value

The updated hv_consort_tracker (invisibly, so it is pipe-safe).

See Also

[hv_consort_start\(\)](#), [hv_consort\(\)](#)

Examples

```
cohort <- data.frame(
  mrn = paste0("P", 1:100),
  age = sample(15:80, 100, TRUE),
  echo = sample(c(TRUE, FALSE), 100, TRUE, prob = c(0.9, 0.1))
)

tracker <- hv_consort_start(cohort, patient_id = mrn) |>
  hv_consort_exclude(
    label = "Eligible",
    col = "excl_screen",
    age < 18 ~ "Age < 18"
  ) |>
  hv_consort_exclude(
    label = "Analyzed",
    col = "excl_eligible",
    !echo ~ "Missing echocardiogram"
  )
```

hv_consort_patients	<i>Retrieve patient IDs at a CONSORT stage</i>
---------------------	--

Description

Returns the IDs of patients who are active at a given stage (or who were excluded for a specific reason at a given stage).

Usage

```
hv_consort_patients(tracker, stage, reason = NULL)
```

Arguments

tracker	An hv_consort_tracker.
stage	Character: either the include_col name (e.g. "eligible") or the stage label (case-insensitive, e.g. "Eligible").
reason	Optional character. If supplied, returns patients excluded from <i>this</i> stage for the specified reason (the string must exactly match a value in the exclusion column). The stage arg then refers to the stage <i>before</i> the exclusion (e.g. "screened" for excl_screen).

Value

A character vector of patient IDs (from the column named in tracker\$patient_id_col).

See Also

[hv_consort_start\(\)](#), [hv_consort_exclude\(\)](#), [hv_consort_summary\(\)](#)

Examples

```
tracker <- hv_consort_start(data.frame(id = 1:10, age = c(rep(15,3), rep(30,7))),
                           patient_id = id) |>
  hv_consort_exclude(label = "Eligible", col = "excl_screen",
                    age < 18 ~ "Age < 18")
hv_consort_patients(tracker, "eligible")
hv_consort_patients(tracker, "screened", reason = "Age < 18")
```

hv_consort_start	<i>Initialize a CONSORT patient-flow tracker</i>
------------------	--

Description

Creates an `hv_consort_tracker` object with one row per patient and a boolean column indicating that every patient is in the initial (screened) population. Build the tracker incrementally with [hv_consort_exclude\(\)](#), then convert to a diagram with [hv_consort\(\)](#).

Usage

```
hv_consort_start(data, patient_id, label = "Screened", pass_col = NULL)
```

Arguments

<code>data</code>	A data frame, one row per patient.
<code>patient_id</code>	<data-masking> Unquoted name of the unique patient identifier column.
<code>label</code>	Character label for the initial population box. Default "Screened".
<code>pass_col</code>	Column name for the initial boolean column. Defaults to <code>ct_snakify(label)</code> (e.g. "screened" when <code>label = "Screened"</code>).

Value

An `hv_consort_tracker` object, a list with:

`$data` Patient-level data frame with boolean/character columns appended per stage.
`$stages` Ordered list of stage descriptors (`label`, `include_col`, `excl_col`, `excl_label`).
`$patient_id_col` Column name of the patient identifier.

See Also

[hv_consort_exclude\(\)](#), [hv_consort\(\)](#)

Examples

```
cohort <- data.frame(mrn = paste0("P", 1:100), age = sample(15:80, 100, TRUE))
tracker <- hv_consort_start(cohort, patient_id = mrn)
print(tracker)
```

hv_consort_summary	<i>Stage-level CONSORT summary table</i>
--------------------	--

Description

Returns a data frame with one row per stage showing patient counts and the exclusion column name, suitable for a methods-section table.

Usage

```
hv_consort_summary(tracker)
```

Arguments

tracker An hv_consort_tracker.

Value

A data frame with columns label, include_col, n_included, excl_col, n_excluded. n_excluded and excl_col are NA for the final stage (no downstream exclusion defined yet).

See Also

[hv_consort_patients\(\)](#)

Examples

```
hv_consort_summary(sample_consort_data())
```

 hv_correlation_matrix *Prepare a scatter-plot matrix*

Description

The scatter-plot-matrix analogue of SAS `proc corr ... plots=matrix`: one scatter panel per unordered pair of vars, lower triangle only. It is the plot half of the `dc-tables` job in the `hvtiR` job catalog. Call `plot()` on the result for a bare `ggplot`; the coefficient table with Fisher intervals is `hvtiRtables::hv_correlation_table()`.

Usage

```
hv_correlation_matrix(
  data,
  vars,
  labels = NULL,
  method = c("pearson", "spearman")
)
```

Arguments

<code>data</code>	Data frame; one row per patient.
<code>vars</code>	Character vector of at least two numeric columns, in display order.
<code>labels</code>	Optional character vector of display labels, one per vars.
<code>method</code>	Coefficient stored in <code>\$tables\$coefficients</code> : "pearson" or "spearman".

Details

Rows missing either coordinate are dropped panel by panel (pairwise deletion, as `proc corr` does), so a variable with missing values thins only the panels it appears in.

A single `warning()` names variables with fewer than 3 non-missing values or no variation, and otherwise-usable pairs with fewer than 3 pairwise-complete observations. These panels are sparse or flat, so their coefficients may be unstable or NA.

At large sizes (about 73 MB at 17 variables by 11,000 rows), write a raster format such as PNG rather than PDF, since every point is a vector object in a PDF, and lower alpha so overplotted panels stay legible.

Value

An object of class `c("hv_correlation_matrix", "hv_data")`: `$data` (columns `col_var`, `row_var`, `x`, `y`), `$meta` (`vars`, `labels`, `method`, `n_obs`, which is `nrow(data)` before any pairwise deletion, not a per-panel count), and `$tables$coefficients`, the pairwise coefficient matrix.

References

SAS template: `descriptive/dc.tables.ods.sas`.

See Also

[plot.hv_correlation_matrix\(\)](#), [sample_correlation_data\(\)](#)

Examples

```
d <- sample_correlation_data()
cm <- hv_correlation_matrix(d, c("a1c", "glucose", "creatinine", "albumin"))
plot(cm) + theme_hv_manuscript()
# strip.placement moves the variable names outside the axis
plot(cm) + ggplot2::theme(strip.placement = "outside")
```

hv_eda

Prepare EDA data for a single variable

Description

Classifies `y_col` using [eda_classify_var](#), pre-processes categorical levels (adding an explicit "(Missing)" level), and returns an `hv_eda` object. Call [plot.hv_eda](#) on the result to obtain a bare `ggplot2` barplot or scatter plot that you can decorate with color scales and [theme_hv_manuscript](#).

Usage

```
hv_eda(
  data,
  x_col = "year",
  y_col = "ef",
  y_label = NULL,
  unique_limit = 6L,
  unique_bound = 100,
  type_overrides = NULL,
  show_percent = FALSE
)
```

Arguments

<code>data</code>	Data frame; one row per observation.
<code>x_col</code>	Name of the reference (time/grouping) column. Used as the x-axis for both scatter and bar plots. Default "year".
<code>y_col</code>	Name of the variable to plot. Default "ef".
<code>y_label</code>	Optional human-readable label for the variable, used as the plot title, y-axis label (continuous), and fill-legend name (categorical). When NULL (default), <code>y_col</code> is used.
<code>unique_limit</code>	Integer threshold passed to eda_classify_var to distinguish categorical from continuous numeric columns. Default 6.
<code>unique_bound</code>	Integer threshold passed to eda_classify_var to distinguish categorical from continuous numeric columns. Default 100.

`type_overrides` Vector of manually set classifications passed to `eda_classify_var`. Default NULL

`show_percent` Logical; for categorical plots, use proportions (`position = "fill"`) instead of counts (`position = "stack"`)? Default FALSE.

Details

Iterate over variables with `lapply()` after selecting columns with `eda_select_vars`.

Value

An object of class `c("hv_eda", "hv_data")`:

`$data` Pre-processed data frame ready for plotting. For continuous variables: two columns `x` and `y`.
For categorical variables: columns `x` (factor) and `fill` (factor with "(Missing)" as explicit level).

`$meta` Named list: `x_col`, `y_col`, `y_label`, `var_type` ("Cont", "Cat_Num", or "Cat_Char"), `show_percent`, `n_obs`.

`$tables` For continuous variables: `rug_data`, the rows where `y_col` is NA, used for the rug layer.
Empty list for categorical variables.

References

R templates: `tp.dp.EDA_barplots_scatterplots.R`, `tp.dp.EDA_barplots_scatterplots_varnames.R`;
helper: `Barplot_Scatterplot_Function.R` (`Function_DataPlotting()`, `Order_Variables()`).

See Also

Worked recipe with rendered output: <https://ehrlinger.github.io/hvtiGraphics/bar.html>.
[plot.hv_eda](#), [sample_eda_data](#), [eda_classify_var](#), [eda_select_vars](#)

Examples

```
dta <- sample_eda_data(n = 300, seed = 42)

# 1. Build data object (binary categorical)
ed <- hv_eda(dta, x_col = "year", y_col = "male", y_label = "Sex")
ed # prints var_type and observation count

# 2. Bare plot -- undecorated ggplot returned by plot.hv_eda
p <- plot(ed)

# 3. Decorate: fill palette, x-axis breaks, labels, theme
p +
  ggplot2::scale_fill_manual(
    values = c("0" = "steelblue", "1" = "firebrick", "(Missing)" = "gray80"),
    labels = c("0" = "Female", "1" = "Male", "(Missing)" = "Missing"),
    name = NULL
  ) +
  ggplot2::scale_x_discrete(breaks = seq(2005, 2020, 5)) +
```

```

ggplot2::labs(x = "Surgery Year", y = "Count") +
  theme_hv_poster()

# Continuous variable -- same 3-step pattern
ed2 <- hv_eda(dta, x_col = "op_years", y_col = "ef",
              y_label = "Ejection Fraction (%)")
plot(ed2) +
  ggplot2::scale_color_manual(values = c("firebrick"), guide = "none") +
  ggplot2::scale_x_continuous(breaks = seq(0, 15, 5)) +
  ggplot2::labs(x = "Years from First Surgery Year") +
  theme_hv_poster()

# Variable selection + lapply (varnames template pattern)
cont_vars <- c(ef = "Ejection Fraction (%)",
               lv_mass = "LV Mass Index (g/m\u00b2)",
               peak_grad = "Peak Gradient (mmHg)")
sub_cont <- eda_select_vars(dta, c("op_years", names(cont_vars)))
p_cont <- lapply(names(cont_vars), function(cn) {
  plot(hv_eda(sub_cont, x_col = "op_years", y_col = cn,
              y_label = cont_vars[[cn]])) +
    ggplot2::scale_color_manual(values = c("steelblue"), guide = "none") +
    ggplot2::scale_x_continuous(breaks = seq(0, 15, 5)) +
    ggplot2::labs(x = "Years from First Surgery Year") +
    theme_hv_poster()
})
p_cont[[1]]

```

Description

Classifies each variable with `eda_classify_var()`, keeps the ones that belong to section, and prepares one `hv_eda()` panel for each. Call `plot.hv_eda_pages()` on the result to lay the panels out as pages.

Usage

```

hv_eda_pages(
  data,
  x_col = "year",
  section = c("continuous", "percent", "count"),
  vars = NULL,
  labels = NULL,
  unique_limit = 6L,
  unique_bound = 100,
  type_overrides = NULL
)

```

Arguments

<code>data</code>	Data frame; one row per patient.
<code>x_col</code>	Name of the reference column, usually the year of operation. Default "year".
<code>section</code>	One of "continuous", "percent" or "count".
<code>vars</code>	Character vector of the variables to consider, in page order. NULL (the default) means every column except <code>x_col</code> and the patient identifiers <code>ccfid</code> , <code>MRN</code> and <code>eMRN</code> (ignoring case), the names <code>hvtiRtemplates</code> ' data contract uses. No other name is guessed at, so <code>PatientID</code> or <code>mrn_num</code> is drawn. An identifier is not a study variable: drawn, it is a scatter of row numbers or a bar chart with one level per patient. <code>meta\$ignored</code> lists what was left out; name a column here to draw it. Every name is checked at once, and the error lists all that are missing.
<code>labels</code>	Optional named character vector of display labels, names being column names. A variable without a label is shown by its name.
<code>unique_limit</code> , <code>unique_bound</code> , <code>type_overrides</code>	Passed to <code>eda_classify_var()</code> .

Details

The three sections split one dataset by variable class:

- "continuous": variables classified "Cont", drawn as scatter plots against `x_col`.
- "percent": categorical variables ("Cat_Num", "Cat_Char"), drawn as bars filled to 100% in each bin of `x_col`.
- "count": the same categorical variables, drawn as stacked counts.

The percent and count sections use the same bins. Categorical panels need a discrete `x`, so `x_col` is binned once, here, and every categorical panel reads that bin. A whole-valued `x_col` (a calendar year) is used as it is. A fractional one (years since an origin) is binned by `floor()`, so each bar is one whole year, and `meta$x_binned` records that it was. Continuous panels always use `x_col` unbinned.

Value

An object of class `c("hv_eda_pages", "hv_data")`:

`$data` One row per variable in the section, in page order: variable, label, var_type.

`$meta` Named list: `x_col`, `section`, `x_binned`, `n_vars`, `n_obs`, `n_other`, the number of considered variables that belong to other sections, and ignored, the identifier columns `vars = NULL` left out.

`$tables` panels, a named list of `hv_eda()` objects, one per variable.

See Also

`plot.hv_eda_pages()`, `hv_eda()`, `eda_classify_var()`

Examples

```
dta <- sample_eda_data(n = 300, seed = 42)

# Continuous variables, four panels to a page
cont <- hv_eda_pages(dta, x_col = "year", section = "continuous")
cont
pages <- plot(cont, ncol = 2, nrow = 2)
length(pages)
attr(pages[[1]], "variables")

# Categorical variables as percentages, decorated across the whole page
pct <- hv_eda_pages(dta, x_col = "year", section = "percent",
  labels = c(male = "Male", nyha = "NYHA class"))
plot(pct)[[1]] &
  ggplot2::scale_fill_manual(values = hv_palette(), na.value = "gray80") &
  theme_hv_manuscript(base_size = 8)
```

hv_fit_location

Fit a plot's chrome on the slide around a fixed panel

Description

Anchors the plot's panel to `panel_box` with `hv_ph_location()` and checks that the whole figure, panel plus the axis text, axis titles, facet strips and plot titles around it, lies on the slide and below the slide title. If it does not, every text size in the plot's theme is reduced by step (as a fraction of the original size), together with every text margin set in absolute units, until it fits or reaches `min_scale`. The panel never moves; only the chrome shrinks.

Usage

```
hv_fit_location(
  plot,
  panel_box,
  slide_width = 13.333,
  slide_height = 7.5,
  title_bottom = 1.38,
  edge = 0.05,
  min_scale = 12/32,
  step = 2/32,
  align = c("panel", "fill"),
  margin = 0.45
)
```

Arguments

<code>plot</code>	A ggplot or patchwork object.
<code>panel_box</code>	Named list <code>list(width, height, left, top)</code> , in inches: the slide rectangle for the plot's panel. See <code>hv_panel_box()</code> .

slide_width, slide_height	Slide size, in inches. The defaults are the 16:9 HVTI templates.
title_bottom	The lowest point of the slide title, in inches. The figure must start below it. The HVTI title placeholder ends at 1.33 in; the default leaves 0.05 in under it.
edge	Margin kept clear on the left, right and bottom slide edges, in inches.
min_scale	Smallest text scale tried, as a fraction of the plot's own sizes. The default, 12/32, is 12 pt text at the 32 pt PPT base size.
step	Amount the scale drops at each try. The default, 2/32, is 2 pt at the 32 pt PPT base size.
align	"panel" (the default) keeps the panel on panel_box, so every slide puts its panel in the same place. "fill" keeps the panel's height and top but sets its left edge and width from the chrome measured around it, so the whole figure spans the slide with margin clear on each side and sits centered. A plot with short y labels then gets a wider panel instead of a gap on the left; one with long labels keeps about the standard width. Panels no longer line up across slides. Text too wide for the margins (a long title) is stepped down as fit steps down any other overflow.
margin	Space kept clear on the left and right of the figure, in inches, with align = "fill". The default, 0.45, matches the right margin of the standard panel box.

Details

At the PPT themes' 32 pt base size the defaults step the text down 2 pt at a time, to 30, 28, 26 and so on, down to 12 pt.

The figure checked is the placement box plus the actual extent of the title, subtitle and caption text, placed by its hjust (and plot.title.position): a title wider than the slide fails the check even though its cell fits.

Only sizes and margins set in the theme are scaled. The size of text drawn by a geom (geom_text(), geom_label()) is an aesthetic, not part of the theme, and is left alone. For a patchwork, panel_box holds the union of its panels and the whole composition is measured; see [hv_ph_location\(\)](#).

Value

A list with elements

- plot: the plot, its text sizes and margins scaled by scale;
- location: the [hv_ph_location\(\)](#) result for panel_box, a list width, height, left, top in inches, ready for [officer::ph_location\(\)](#);
- scale: the text scale used, 1 when the plot fit as it was;
- fits: TRUE when the figure lies inside the slide limits. When it is FALSE, plot and location are those at min_scale.
- panel_box: the panel rectangle used, panel_box itself unless align = "fill" moved it.

See Also

[hv_add_plot\(\)](#), which places the result on a slide; [hv_panel_box\(\)](#), [hv_ph_location\(\)](#).

Examples

```
p <- ggplot2::ggplot(mtcars, ggplot2::aes(hp, mpg)) +
  ggplot2::geom_point() +
  ggplot2::labs(x = "Horsepower", y = "Miles per gallon") +
  theme_hv_ppt_dark()
fit <- hv_fit_location(p, hv_panel_box("dark"))
fit$scale
fit$location
```

hv_followup

Prepare goodness-of-follow-up data for plotting

Description

Validates dates, builds per-patient follow-up frames, and returns an `hv_followup` object containing the data for both the death panel (`type = "followup"`) and, optionally, the event panel (`type = "event"`). Call `plot.hv_followup` on the result to obtain a bare `ggplot2` object.

Usage

```
hv_followup(
  data,
  iv_opyrs_col = "iv_opyrs",
  death_col = "dead",
  death_time_col = "iv_dead",
  event_col = NULL,
  event_time_col = NULL,
  death_for_event_col = NULL,
  origin_year = 1990,
  study_start = as.Date("1990-01-01"),
  study_end = as.Date("2019-12-31"),
  close_date = as.Date("2021-08-06"),
  tolower_names = TRUE,
  death_levels = c("Alive", "Dead"),
  event_levels = c("No event", "Non-fatal event", "Death"),
  segment_drop = 0
)
```

Arguments

<code>data</code>	A data frame with one row per patient. See sample_goodness_followup_data .
<code>iv_opyrs_col</code>	Name of the operation-year column. Default <code>"iv_opyrs"</code> .
<code>death_col</code>	Name of the binary death-indicator column. Default <code>"dead"</code> .
<code>death_time_col</code>	Name of the time-to-death column. Default <code>"iv_dead"</code> .

event_col	Name of the non-fatal event indicator column. Required to compute the event panel (type = "event"). Default NULL (event panel unavailable). The flag alone does not decide the state: the event time is compared against death_time_col, and a flagged event is only counted as non-fatal when it strictly precedes death. Ties go to death.
event_time_col	Name of the time-to-event column. Required when event_col is supplied. Default NULL.
death_for_event_col	Name of the death column to use specifically in the event panel (defaults to death_col when NULL).
origin_year	Integer; calendar year used as the y-axis origin. Default 1990.
study_start	Start of study period. Default as.Date("1990-01-01").
study_end	End of study enrollment. Default as.Date("2019-12-31").
close_date	Data close date. Must be $\geq$ study_end. Default as.Date("2021-08-06").
tolower_names	Logical; if TRUE, column names are lower-cased when materialising the data. Default TRUE.
death_levels	Length-2 character vector labeling the two death states (alive first). Default c("Alive", "Dead").
event_levels	Length-3 character vector for the event panel (event-free, non-fatal event, death). Default c("No event", "Non-fatal event", "Death").
segment_drop	Numeric; vertical offset (years) for the segment endpoint below the follow-up point. Default 0 (no segment drawn). Set to a positive value to restore the legacy stem below each point.

Value

An object of class c("hv_followup", "hv_data"):

`$data` Per-patient data frame for the death panel.

`$meta` Column names, date parameters, state levels, has_event flag, and the cohort counts `n_patients` (analyzed), `n_input`, and `n_excluded`.

`$tables` Named list with diagonal (the study-period reference diagonal) and, when event columns are supplied, `event_data`.

Missing data

Patients with a missing value in any required column are excluded, with a warning naming the columns responsible. `$meta$n_patients` reports the *analyzed* cohort, alongside `$meta$n_input` and `$meta$n_excluded`, so the reported N always describes the plotted points.

The event panel requires more columns than the death panel, so the two can have different cohorts: a patient with a complete death record but a missing event time appears in one panel and not the other. Each panel is therefore filtered and warned about separately, and the event panel's counts are reported as `$meta$n_event_patients` and `$meta$n_event_excluded`.

See Also

Worked recipe with rendered output: <https://ehrlinger.github.io/hvtiGraphics/balance.html>.

```
plot.hv_followup, sample_goodness_followup_data
```

Examples

```
dta <- sample_goodness_followup_data()

# 1. Build data object
gf <- hv_followup(dta)
gf # prints follow-up summary

# 2. Bare plot -- undecorated ggplot returned by plot.hv_followup
p <- plot(gf)

# 3. Decorate: color palette, axis labels, theme
p +
  ggplot2::scale_color_manual(
    values = c("Alive" = "steelblue", "Dead" = "firebrick"),
    name = NULL
  ) +
  ggplot2::labs(x = "Operation Date", y = "Follow-up (years)") +
  theme_hv_poster()

# With event panel -- same 3-step pattern
gf2 <- hv_followup(dta, event_col = "ev_event", event_time_col = "iv_event")
plot(gf2, type = "event") +
  ggplot2::scale_color_manual(
    values = c("No event" = "blue", "Non-fatal event" = "green3",
              "Death" = "red"),
    name = NULL
  ) +
  ggplot2::scale_shape_manual(values = c(1, 2, 4), name = NULL) +
  ggplot2::labs(x = "Operation Date", y = "Follow-up (years)") +
  theme_hv_poster()
```

hv_followup_panels	<i>Prepare a set of goodness-of-follow-up panels</i>
--------------------	--

Description

Checks a set of death and event panels against the data, works out the study window they share, and prepares one `hv_followup()` object per panel. Call `plot.hv_followup_panels()` on the result for the figures.

Usage

```

hv_followup_panels(
  data,
  opyrs_col = "iv_opyrs",
  origin_year,
  panels,
  events = list(),
  close_date = NULL
)

```

Arguments

data	Data frame; one row per patient.
opyrs_col	Name of the years-since-origin interval to the operation. Default "iv_opyrs".
origin_year	The calendar year opyrs_col counts from, a whole number.
panels	Named list of death panels. Each entry is a list with status, the 1/0 death indicator, time, its follow-up in years, and optionally title.
events	Named list of non-fatal event panels, possibly empty. Each entry is a list with event and time, the event's indicator and interval, death and death_time, the death it competes with, and optionally label. A flagged event counts only when it strictly precedes death.
close_date	NULL to estimate, or one date (a Date or a string as .Date() reads), on or after the last operation.

Details

The study window starts on 1 January of origin_year, because `hv_followup()` draws its potential-follow-up diagonal from origin_year whatever study_start is. It ends at the latest operation in the data.

The close date is close_date when given. When NULL it is estimated as the latest operation plus follow-up in the data: the date follow-up is known to reach, which is not necessarily the date it was closed. meta\$close_source says which, so a report can too.

Every check runs before any panel is built. Every column named anywhere is checked at once, and the error lists all that are missing. A panel name is also its figure's name, so a name used twice, within panels or across panels and events, is an error. The origin is checked for plausibility: operation years outside 1900 to next year are the wrong-origin mistake, which otherwise slides every point along the x-axis without any other symptom, and stop.

An operation before the origin, a negative opyrs_col, is drawn where it falls, to the left of where the diagonal starts, with a warning, and counted in meta\$n_opyrs_negative. It usually means the build counted opyrs_col from a later origin than origin_year, and a figure that shows the affected patients makes that a problem a reader can see, where a refusal would hide the figure and the patients with it. When **every** operation is before the origin, the origin is wrong for the whole cohort and there is no window to draw, so that stops.

Value

An object of class `c("hv_followup_panels", "hv_data")`:

`$data` One row per panel, in order, death panels first: `panel`, `type` ("followup" or "event"), `title`, `n_drawn` and `n_excluded`.

`$meta` Named list: `opysr_col`, `origin_year`, `study_start`, `study_end`, `first_operation`, `close_date`, `close_source`, `n_obs`, `n_opysr_missing` and `n_opysr_negative`.

`$tables` `panels`, a named list of `hv_followup()` objects.

See Also

`plot.hv_followup_panels()`, `hv_followup()`

Examples

```
dta <- sample_goodness_followup_data()
fp <- hv_followup_panels(
  dta, origin_year = 1990,
  panels = list(all = list(status = "dead", time = "iv_dead", title = "All deaths")),
  events = list(ev = list(event = "ev_event", time = "iv_event", death = "deads",
    death_time = "iv_dead", label = "Non-fatal event"))
)
fp
fp$meta$close_source
```

hv_ggsave_dims	Compute ggsave() dimensions for a fixed panel content area
----------------	--

Description

Given a fitted ggplot, returns width and height values such that, when passed to `ggplot2::ggsave()`, the resulting file has a panel content area matching width / height. The "panel content area" is the rectangular bounding box of the `gtable` cells tagged `panel`, i.e., the smallest rectangle that encloses every plotting panel. Whatever grobs fall inside that rectangle (e.g., inter-panel gutters and strip rows that sit between facet rows) are counted as part of the target; everything outside it (axes, axis titles, plot title, caption, legend, plot margins, and any strips that sit above/below or beside the panel block such as `facet_grid` side strips) is counted as chrome.

Usage

```
hv_ggsave_dims(plot, width, height, units = c("in", "cm", "mm"))
```

Arguments

- `plot` A ggplot (or patchwork) object.
- `width` Target panel content area width, in units.
- `height` Target panel content area height, in units.
- `units` One of "in", "cm", "mm". Default "in".

Details

Useful for multi-panel figure sets where a constant data region is required across PDFs regardless of label length or legend placement. Always measured with a PDF sizing device, which is why units is limited to length (inches, cm, mm); DPI is irrelevant for vector output.

Value

A named list with elements width, height, units, shaped to splat directly into `ggplot2::ggsave()` via `do.call()` (see examples).

See Also

`hv_ph_location()` for the slide-placement analogue used with `save_ppt()`.

Examples

```
p <- ggplot2::ggplot(mtcars, ggplot2::aes(hp, mpg)) +
  ggplot2::geom_point() +
  ggplot2::labs(title = "Long title that eats vertical space",
               x = "Horsepower", y = "Miles per gallon")

dims <- hv_ggsave_dims(p, width = 4, height = 3)
do.call(
  ggplot2::ggsave,
  c(list(filename = file.path(tempdir(), "fig.pdf"), plot = p), dims)
)
```

 hv_girafe

Render a plot or page as an interactive HTML widget

Description

Wraps `ggiraph::girafe()` for EDA pages drawn with `interactive = TRUE` (see `plot.hv_eda()`), so hovering a bar shows its tooltip.

Usage

```
hv_girafe(x, width = 6.5, height = 9, ...)
```

Arguments

<code>x</code>	A ggplot or patchwork object.
<code>width, height</code>	Size of the drawing in inches. Default 6.5 by 9, the printed page.
<code>...</code>	Passed to <code>ggiraph::girafe()</code> , for example options.

Details

No fonts are embedded. By default ggiraph embeds its font files in the widget, which made one 8-panel page 8.6 MB instead of 0.4 MB. Here text is set in Arial (or the nearest system sans serif) and the browser draws it in the reader's own copy. Saved PNG and PDF output is unaffected.

Value

An htmlwidget of class girafe. Printed in a knitr or Quarto HTML document, it is drawn in place.

See Also

[plot.hv_eda\(\)](#), [plot.hv_eda_pages\(\)](#)

Examples

```
if (requireNamespace("ggiraph", quietly = TRUE)) {
  dta <- sample_eda_data(n = 300, seed = 42)
  pages <- plot(hv_eda_pages(dta, section = "percent"),
               ncol = 2, nrow = 2, interactive = TRUE)
  widget <- hv_girafe(pages[[1]], width = 6.5, height = 6)
  class(widget)
}
```

hv_hazard

Prepare parametric hazard / survival data for plotting

Description

Validates and stores pre-computed parametric curve data (and optional Kaplan-Meier empirical overlay and population life-table reference) as an hv_hazard object. Pass the result to [plot.hv_hazard\(\)](#) to render the figure.

Usage

```
hv_hazard(
  curve_data,
  x_col = "time",
  estimate_col = "survival",
  lower_col = NULL,
  upper_col = NULL,
  group_col = NULL,
  empirical = NULL,
  emp_x_col = x_col,
  emp_estimate_col = "estimate",
  emp_lower_col = NULL,
  emp_upper_col = NULL,
  emp_group_col = group_col,
  emp_geom = c("point", "step"),
```

```

    reference = NULL,
    ref_x_col = x_col,
    ref_estimate_col = estimate_col,
    ref_group_col = NULL
  )

```

Arguments

curve_data	Data frame of parametric predictions (fine grid). Typical source: SAS predict dataset exported to CSV, or sample_hazard_data() .
x_col	Name of the time / age column. Default "time".
estimate_col	Name of the predicted-value column: "survival", "hazard", or "cumhaz". Default "survival".
lower_col	Lower CI column in curve_data, or NULL for no ribbon. Default NULL.
upper_col	Upper CI column in curve_data, or NULL. Default NULL.
group_col	Stratification column in curve_data, or NULL for a single curve. Default NULL.
empirical	Optional data frame of KM empirical points (SAS plout dataset). Stored in \$tables\$empirical. Default NULL.
emp_x_col	x column in empirical. Defaults to x_col.
emp_estimate_col	y column in empirical. Default "estimate".
emp_lower_col	Lower error-bar column in empirical, or NULL. Default NULL.
emp_upper_col	Upper error-bar column in empirical, or NULL. Default NULL.
emp_group_col	Group column in empirical. Defaults to group_col.
emp_geom	Geom used for the empirical overlay: "point" (default, open circles) or "step" (Kaplan-Meier step function).
reference	Optional data frame of population life-table curves (SAS smatched dataset). Stored in \$tables\$reference. Default NULL.
ref_x_col	x column in reference. Defaults to x_col.
ref_estimate_col	y column in reference. Defaults to estimate_col.
ref_group_col	Linetype-grouping column in reference, or NULL. Default NULL.

Details

This constructor covers the complete tp.hp.dead.* SAS template family. All column-name mappings and all three data frames are fixed at construction time; aesthetic parameters (ci_alpha, line_width, etc.) are passed to plot().

Value

An S3 object of class c("hv_hazard", "hv_data"); call plot() on the result to render (see [plot.hv_hazard\(\)](#)). The object contains: \$data (curve data frame), \$meta (all column-name mappings), \$tables\$empirical, \$tables\$reference.

See Also

Worked recipe with rendered output: <https://ehrlinger.github.io/hvtiGraphics/hazard.html>.

`plot.hv_hazard()` to render as a ggplot2 figure, `theme_hv_manuscript()` for the publication theme, `sample_hazard_data()`, `sample_hazard_empirical()`, `sample_life_table()` for example data.

Other Hazard plot: `plot.hv_hazard()`

Examples

```
library(ggplot2)

dat <- sample_hazard_data(n = 500, time_max = 10)
emp <- sample_hazard_empirical(n = 500, time_max = 10, n_bins = 6)

# 1. Build data object
hp <- hv_hazard(dat,
  lower_col = "surv_lower", upper_col = "surv_upper",
  empirical = emp,
  emp_lower_col = "lower", emp_upper_col = "upper"
)
hp # prints CI and empirical flags

# 2. Bare plot -- undecorated ggplot returned by plot.hv_hazard
p <- plot(hp)

# 3. Decorate: color/fill palettes, axis scales, labels, theme
p +
  scale_color_manual(values = c("steelblue"), guide = "none") +
  scale_fill_manual(values = c("steelblue"), guide = "none") +
  scale_x_continuous(limits = c(0, 10), breaks = 0:10) +
  scale_y_continuous(limits = c(0, 100), breaks = seq(0, 100, 20),
    labels = function(x) paste0(x, "%")) +
  labs(x = "Years", y = "Survival (%)") +
  theme_hv_poster()

# Stratified groups -- color scale adds clinical meaning
dat2 <- sample_hazard_data(
  n = 400, groups = c("No Takedown" = 1.0, "Takedown" = 0.65)
)
hp2 <- hv_hazard(dat2,
  lower_col = "surv_lower", upper_col = "surv_upper",
  group_col = "group"
)
plot(hp2) +
  scale_color_manual(
    values = c("No Takedown" = "steelblue", "Takedown" = "firebrick"),
    name = NULL
  ) +
  labs(x = "Years", y = "Survival (%)") +
  theme_hv_poster()
```

```
# --- Global theme + RColorBrewer (set once per session) -----

old <- ggplot2::theme_set(theme_hv_manuscript())
plot(hp2) +
  scale_color_manual(values = hv_palette(), name = NULL) +
  scale_fill_manual(values = hv_palette(), guide = "none") +
  scale_x_continuous(limits = c(0, 10), breaks = 0:10) +
  scale_y_continuous(limits = c(0, 100), breaks = seq(0, 100, 20),
    labels = function(x) paste0(x, "%")) +
  labs(x = "Years", y = "Survival (%)")
ggplot2::theme_set(old)

# See vignette("plot-decorators", package = "hvtiPlotR") for theming,
# color scales, annotation labels, and saving plots.
```

hv_legend_inside	<i>Place a ggplot legend inside the emptiest panel corner</i>
------------------	---

Description

Inspects a built plot, finds the panel corner with the fewest data points, and anchors the legend there. When no corner is clear of data (e.g. dense multi-curve panels), or the plot is faceted, the legend is sent to an outside position instead. The CORR house convention is in-panel legends placed in dead space; this automates the corner choice.

Usage

```
hv_legend_inside(
  plot,
  threshold = 0.08,
  box_frac = 0.3,
  pad = 0.02,
  fallback = "right",
  prefer = NULL
)
```

Arguments

plot	A single-panel ggplot.
threshold	Maximum fraction of data points allowed in the chosen corner box for an in-panel legend. If the emptiest corner exceeds it, fallback is used. Default 0.08.
box_frac	Corner-box size as a fraction of panel width and height (≤ 0.5). Default 0.30.
pad	Inset of the legend anchor from the panel edge, in npc units. Default 0.02.

fallback	Outside legend.position used when no corner is empty enough or the plot cannot be reasoned about (facets). One of "right", "left", "top", "bottom". Default "right".
prefer	Optional preferred corner: one of "topright", "topleft", "bottomright", "bottomleft". When set and that corner is clear (its occupancy is within threshold), the legend is placed there even if another corner is emptier. If the preferred corner is occupied, the emptiest-corner logic applies as usual. Default NULL (pick the emptiest corner).

Details

Apply it *after* the house theme (which sets legend.position = "none") so its position wins.

Value

The input plot with a `theme` layer added that sets the legend position.

See Also

Worked recipe with rendered output: <https://ehrlinger.github.io/hvtiGraphics/legends.html>.

Examples

```
if (requireNamespace("ggplot2", quietly = TRUE)) {
  library(ggplot2)
  p <- ggplot(mtcars, aes(wt, mpg, color = factor(cyl))) + geom_point()
  hv_legend_inside(p)
}
```

hv_longitudinal

Prepare longitudinal participation counts data for plotting

Description

Validates a pre-aggregated long-format counts data frame and returns an hv_longitudinal object. Call `plot.hv_longitudinal` on the result with type = "plot" for the grouped bar chart or type = "table" for the numeric text-table panel. Compose both panels with **patchwork**.

Usage

```
hv_longitudinal(
  data,
  x_col = "time_label",
  count_col = "count",
  group_col = "series"
)
```

Arguments

data	Long-format data frame; one row per series per time point. See sample_longitudinal_counts_data .
x_col	Name of the discrete time-label column. Default "time_label".
count_col	Name of the numeric count column. Default "count".
group_col	Name of the series grouping column. Default "series".

Value

An object of class `c("hv_longitudinal", "hv_data")`:

`$data` The validated input data frame.

`$meta` Named list: `x_col`, `count_col`, `group_col`, `n_timepoints`, `n_groups`, `n_obs`.

`$tables` Empty list.

See Also

Worked recipe with rendered output: <https://ehrlinger.github.io/hvtiGraphics/bar.html>.
[plot.hv_longitudinal](#), [sample_longitudinal_counts_data](#)

Examples

```
dta <- sample_longitudinal_counts_data(n_patients = 300, seed = 42L)

# 1. Build data object
lc <- hv_longitudinal(dta)
lc # prints group/time-point counts

# 2. Bare plot -- undecorated ggplot returned by plot.hv_longitudinal
library(ggplot2)
p <- plot(lc, type = "plot")

# 3. Decorate: fill palette, y-axis scale, labels, theme
p +
  scale_fill_manual(
    values = c(Patients = "steelblue", Measurements = "firebrick"),
    name = NULL
  ) +
  scale_y_continuous(labels = scales::comma,
                     breaks = seq(0, 2000, 500),
                     expand = c(0, 0)) +
  coord_cartesian(ylim = c(0, 2200)) +
  labs(x = "Follow-up Window", y = "Count (n)") +
  theme_hv_poster() +
  theme(legend.position = c(0.85, 0.85))
```

hv_mirror_hist	<i>Prepare mirror-histogram data for plotting</i>
----------------	---

Description

Validates and assembles propensity-score distributions for a mirrored histogram comparing a treated group (bars above the axis) and a control group (bars below the axis), with optional matched and unmatched shading. Returns an `hv_mirror_hist` object; call `plot.hv_mirror_hist` on the result to obtain a bare `ggplot2` object.

Usage

```
hv_mirror_hist(
  data,
  score_col = "prob_t",
  group_col = "tavr",
  match_col = "match",
  group_levels = c(0, 1),
  group_labels = c("SAVR", "TF-TAVR"),
  matched_value = 1,
  score_multiplier = HVTI_SCORE_DEFAULT_MULTIPLIER,
  binwidth = 5,
  weight_col = NULL
)
```

Arguments

<code>data</code>	A data frame with one row per patient.
<code>score_col</code>	Name of the propensity-score column (0–1 scale before multiplier is applied). Default "prob_t".
<code>group_col</code>	Name of the binary group-indicator column. Default "tavr".
<code>match_col</code>	Name of the binary match-indicator column (1 = matched). Default "match".
<code>group_levels</code>	Length-2 vector of the two values in <code>group_col</code> (control first, treated second). Default <code>c(0, 1)</code> .
<code>group_labels</code>	Length-2 character vector of display labels corresponding to <code>group_levels</code> . Default <code>c("SAVR", "TF-TAVR")</code> .
<code>matched_value</code>	Value in <code>match_col</code> that flags a matched patient. Default 1.
<code>score_multiplier</code>	Scalar applied to <code>score_col</code> to convert to the 0–100 display scale. Default 100.
<code>binwidth</code>	Histogram bin width on the 0–100 scale. Default 5.
<code>weight_col</code>	Optional name of an IPTW weight column. When supplied, bar heights reflect weighted counts instead of raw counts.

Value

An object of class `c("hv_mirror_hist", "hv_data")`; call `plot()` on the result to render the figure (see [plot.hv_mirror_hist](#)). The list contains:

`$data` Tidy data frame of histogram bar coordinates for [plot.hv_mirror_hist](#).

`$meta` Named list: `score_col`, `group_col`, `match_col`, `group_labels`, `binwidth`, `lower`, `upper`, `y_breaks`, `n_obs`, `n_dropped`.

`$tables` Named list with two elements:

`diagnostics` A named list of diagnostic summaries. Always contains `n_input`, `n_analyzed`, `n_dropped_missing_or_other_group`, `group_counts_before` (table), `score_summary_before` (by object), and `smd_before` (numeric SMD). In binary-match mode, additionally contains `group_counts_matched`, `matched_rate_by_group`, `score_summary_matched`, and `smd_matched`. In weighted IPTW mode, additionally contains `effective_n_by_group` and `smd_weighted`.

`working` The per-patient data frame after score rescaling and complete-case filtering, used for custom downstream diagnostics.

See Also

Worked recipe with rendered output: <https://ehrlinger.github.io/hvtiGraphics/histograms.html>.

[plot.hv_mirror_hist](#) to render as a ggplot2 figure, [theme_hv_manuscript](#) for the publication theme, [sample_mirror_histogram_data](#) for example data.

Other Propensity Score & Matching: [hv_rmst_contrast\(\)](#), [hv_rmst_curves\(\)](#), [plot.hv_mirror_hist\(\)](#), [plot.hv_rmst_contrast\(\)](#), [plot.hv_rmst_curves\(\)](#), [sample_covariate_balance_data\(\)](#)

Examples

```
dta <- sample_mirror_histogram_data(n = 500, separation = 1.5)

# 1. Build data object
mh <- hv_mirror_hist(dta)
mh                                     # print diagnostics summary
mh$tables$diagnostics                # full diagnostics list

# 2. Bare plot -- undecorated ggplot returned by plot.hv_mirror_hist
p <- plot(mh)

# 3. Decorate: axis labels and theme
p +
  ggplot2::labs(x = "Propensity Score (%)", y = "Count") +
  theme_hv_poster()
```

hv_nnt

*Prepare number-needed-to-treat data for plotting***Description**

Stores pre-computed NNT (or ARR) data as an hv_nnt object. Rows where estimate_col is NA (undefined when ARR is near zero) are optionally removed during construction via na_rm. Pass the result to `plot.hv_nnt()` to render the plot. Covers the NNT component of `tp.hp.numtreat.survdiff.matched.sas`.

Usage

```
hv_nnt(
  nnt_data,
  x_col = "time",
  estimate_col = "nnt",
  lower_col = NULL,
  upper_col = NULL,
  group_col = NULL,
  na_rm = TRUE
)
```

Arguments

nnt_data	Data frame of pre-computed NNT estimates. See <code>sample_nnt_data()</code> .
x_col	Name of the time column. Default "time".
estimate_col	Name of the NNT (or ARR) column. Default "nnt".
lower_col	Lower CI column, or NULL. Default NULL.
upper_col	Upper CI column, or NULL. Default NULL.
group_col	Grouping column for multiple comparisons, or NULL. Default NULL.
na_rm	Remove rows where estimate_col is NA before storing in \$data. Applied at construction time. Default TRUE.

Value

An S3 object of class `c("hv_nnt", "hv_data")`.

See Also

Worked recipe with rendered output: <https://ehrlinger.github.io/hvtiGraphics/nnt.html>.
`plot.hv_nnt()`, `sample_nnt_data()`, `hv_survival_difference()`

Examples

```
library(ggplot2)

nnt_dat <- sample_nnt_data(
  n = 500, time_max = 20,
  groups = c("SVG" = 1.0, "ITA" = 0.75)
)

# 1. Build data object
nn <- hv_nnt(nnt_dat, lower_col = "nnt_lower", upper_col = "nnt_upper")
nn # prints estimate column and CI flags

# 2. Bare plot -- undecorated ggplot returned by plot.hv_nnt
p <- plot(nn)

# 3. Decorate: axis scales, labels, theme
p +
  scale_x_continuous(limits = c(0, 20), breaks = seq(0, 20, 5)) +
  scale_y_continuous(limits = c(0, 50), breaks = seq(0, 50, 10)) +
  labs(x = "Years", y = "Number Needed to Treat (NNT)") +
  theme_hv_poster()

# ARR curve -- same 3-step pattern with a different estimate column
ar <- hv_nnt(nnt_dat, estimate_col = "arr",
             lower_col = "arr_lower", upper_col = "arr_upper",
             na_rm = FALSE)

plot(ar) +
  scale_y_continuous(limits = c(0, 50),
                    labels = function(x) paste0(x, "%")) +
  labs(x = "Years", y = "Absolute Risk Reduction (%)") +
  theme_hv_poster()
```

hv_nonparametric

Prepare nonparametric temporal trend curve data for plotting

Description

Validates pre-computed curve data (and optional CI bounds and binned data summary points) and returns an `hv_nonparametric` object. Call [plot.hv_nonparametric](#) to obtain a bare `ggplot2` curve plot that you can decorate with color/fill scales, axis limits, and [theme_hv_manuscript](#).

Usage

```
hv_nonparametric(
  curve_data,
  x_col = "time",
  estimate_col = "estimate",
  lower_col = NULL,
```

```

    upper_col = NULL,
    group_col = NULL,
    data_points = NULL
  )

```

Arguments

curve_data	Data frame; one row per (time, group) combination.
x_col	Name of the x-axis column. Default "time".
estimate_col	Name of the predicted value column. Default "estimate".
lower_col	Name of the lower CI bound column, or NULL. Default NULL.
upper_col	Name of the upper CI bound column, or NULL. Default NULL.
group_col	Name of the stratification column, or NULL. Default NULL.
data_points	Optional data frame of binned data summary points. Must have columns matching x_col and "value", plus group_col when stratified. Default NULL.

Details

Covers the full range of tp.np.* SAS templates:

SAS template pattern	R usage
Single average curve (avrg_curv, u.trend)	hv_nonparametric(dat)
Curve + 68\ Curve + CI + data points	+ data_points = ...
Two-group comparison (double, ozak)	+ group_col = "group"
Multi-scenario / covariate-adjusted (mult)	+ group_col = "group"
Phase decomposition (phases, independence)	+ group_col = "phase"

SAS column mapping:

- estimate_col <- prev, mnprev, _p_, est_fev, est_z0d
- lower_col <- cll_p68 or cll_p95
- upper_col <- clu_p68 or clu_p95
- group_col <- indicator added after wide-to-long reshape

Value

An object of class c("hv_nonparametric", "hv_data"); call plot() on the result to render the figure; see [plot.hv_nonparametric](#). The list contains:

\$data The curve_data data frame.

\$meta Named list: x_col, estimate_col, lower_col, upper_col, group_col, has_ci, has_data_points, n_obs.

\$tables List; contains data_points when supplied.

See Also

Worked recipe with rendered output: <https://ehrlinger.github.io/hvtiGraphics/hazard.html>.

`plot.hv_nonparametric` to render as a ggplot2 figure, `theme_hv_manuscript` for the publication theme, `sample_nonparametric_curve_data` for example data.

Other Nonparametric curves: `plot.hv_nonparametric()`

Examples

```
dat      <- sample_nonparametric_curve_data(n = 500, time_max = 12)
dat_pts  <- sample_nonparametric_curve_points(n = 500, time_max = 12)

# 1. Build data object
np <- hv_nonparametric(dat, lower_col = "lower", upper_col = "upper",
                      data_points = dat_pts)
np # prints CI / data-point flags

# 2. Bare plot -- undecorated ggplot returned by plot.hv_nonparametric
p <- plot(np)

# 3. Decorate: color/fill palettes, axis scales, labels, theme
p +
  ggplot2::scale_color_manual(values = c("steelblue"), guide = "none") +
  ggplot2::scale_fill_manual(values = c("steelblue"), guide = "none") +
  ggplot2::scale_x_continuous(limits = c(0, 12), breaks = 0:12) +
  ggplot2::scale_y_continuous(limits = c(0, 0.40),
                             breaks = seq(0, 0.40, 0.10),
                             labels = scales::percent) +
  ggplot2::labs(x = "Months", y = "Prevalence of AF") +
  theme_hv_poster()

# --- Global theme (set once per session) -----

old <- ggplot2::theme_set(theme_hv_manuscript())
plot(np) +
  ggplot2::scale_color_manual(values = c("steelblue"), guide = "none") +
  ggplot2::scale_fill_manual(values = c("steelblue"), guide = "none") +
  ggplot2::labs(x = "Months", y = "Prevalence of AF")
# For multi-group curves swap scale_color_manual with:
# ggplot2::scale_color_manual(values = hv_palette(), name = NULL)
# ggplot2::scale_fill_manual(values = hv_palette(), guide = "none")
ggplot2::theme_set(old)

# See vignette("plot-decorators", package = "hvtiPlotR") for theming,
# color scales, annotation labels, and saving plots.
```

hv_ordinal

Prepare nonparametric ordinal outcome curve data for plotting

Description

Validates pre-computed grade-specific probability curves (and optional binned data summary points) and returns an hv_ordinal object. Call `plot.hv_ordinal` on the result to obtain a bare ggplot2 multi-grade line plot that you can decorate with color scales and `theme_hv_manuscript`.

Usage

```
hv_ordinal(
  curve_data,
  x_col = "time",
  estimate_col = "estimate",
  grade_col = "grade",
  data_points = NULL
)
```

Arguments

curve_data	Long-format data frame: one row per (time, grade) combination. Columns: x_col, estimate_col, grade_col.
x_col	Name of the time column. Default "time".
estimate_col	Name of the predicted probability column. Default "estimate".
grade_col	Name of the grade/category column. Default "grade".
data_points	Optional long-format data frame of binned data summary points. Must have columns matching x_col, "value", and grade_col. Default NULL.

Details

SAS column mapping (predict dataset after averaging):

- time <- iv_echo (or iv_wristm)
- estimate <- one of p0, p1, p2, p3 (individual grade probs, after wide-to-long reshape)
- grade <- a new column created during the reshape

Value

An object of class c("hv_ordinal", "hv_data"):

\$data The curve_data data frame.

\$meta Named list: x_col, estimate_col, grade_col, n_obs, n_grades, has_data_points.

\$tables List; contains data_points when supplied.

References

SAS templates: `tp.np.tr.ivecho.average_curv.ordinal.sas`, `tp.np.po_ar.u_multi.ordinal.sas`, `tp.np.tr.ivecho.independence.sas`, `tp.np.tr.ivecho.u.phases.sas`.

See Also

Worked recipe with rendered output: <https://ehrlinger.github.io/hvtiGraphics/hazard.html>.

`plot.hv_ordinal`, `sample_nonparametric_ordinal_data`

Examples

```
dat <- sample_nonparametric_ordinal_data(
  n = 800, time_max = 5,
  grade_labels = c("None", "Mild", "Moderate", "Severe")
)
dat_pts <- sample_nonparametric_ordinal_points(
  n = 800, time_max = 5,
  grade_labels = c("None", "Mild", "Moderate", "Severe")
)

# 1. Build data object
ord <- hv_ordinal(dat, data_points = dat_pts)
ord # prints grade count and data-point flag

# 2. Bare plot -- undecorated ggplot returned by plot.hv_ordinal
p <- plot(ord)

# 3. Decorate: color palette, axis scales, labels, theme
p +
  ggplot2::scale_color_manual(
    values = c(None = "steelblue",
               Mild = "firebrick",
               Moderate = "forestgreen",
               Severe = "goldenrod3"),
    name = "TR Grade"
  ) +
  ggplot2::scale_x_continuous(breaks = 0:5) +
  ggplot2::scale_y_continuous(limits = c(0, 0.50),
                             breaks = seq(0, 0.50, 0.10),
                             labels = scales::percent) +
  ggplot2::labs(x = "Years", y = "Percent in each TR grade") +
  theme_hv_poster()
```

Description

Colors for categorical groups in a manuscript, poster or light-background slide figure: the Okabe-Ito colors, ordered for a white background, with every one clearing 3:1 contrast against white. Hand it to a manual scale, `scale_color_manual(values = hv_palette())`.

Usage

```
hv_palette(n = NULL)
```

Arguments

`n` Number of colors to return, or NULL (default) for all six.

Details

This is the categorical half of the house color rule. Ordered and diverging scales take ColorBrewer "RdBu" or "PuOr" through `ggplot2::scale_color_brewer()`, and sequential scales take "Blues". Avoid "Set1" and "RdYlGn": their red against green fails for a reader with deuteranopia.

The colors are those of `hv_ppt_palette("light")`, under a name that does not tie a manuscript figure to slides. For a dark slide, use `hv_ppt_palette("dark")`. Six colors are supplied, and asking for more is an error rather than a silently recycled palette.

Value

A character vector of hex colors, in group order.

See Also

`hv_ppt_palette()` for the slide orderings, `hv_role_palette()` when levels carry a role such as event or censored.

Examples

```
hv_palette()
hv_palette(n = 3)
```

hv_panel_box

Standard slide panel rectangle for a slide format

Description

The one fixed panel rectangle every plot slide of a slide format anchors its plotting panel to, in inches. The slide format is the slide's aspect ratio: "wide" is 16:9, 13.333 by 7.5 in, the size of every bundled HVTI template; "standard" is 4:3, 10 by 7.5 in. The color template does not matter: the dark, light and slate templates get the same rectangle. `save_ppt()` and `hv_add_plot()` read the format from the template's slide size and use this rectangle by default. Axis text, axis titles, facet strips and plot titles flow outside it; `hv_fit_location()` keeps them on the slide and out of the slide title.

Usage

```

hv_panel_box(
  mode = c("dark", "light", "slate"),
  format = c("wide", "standard")
)

```

Arguments

mode	"dark", "light" or "slate". Kept for backward compatibility: every color template gets the same rectangle, so it has no effect.
format	"wide" (16:9, the default) or "standard" (4:3, provisional).

Details

The wide rectangle is 9.98 by 4.11 in, 2.9 in from the left edge and 1.85 in from the top. The numbers leave room for the chrome around the panel:

- 2.9 in on the left for long y-axis labels and the rotated y-axis title;
- about 0.45 in above the panel, below the title placeholder that ends at 1.33 in, for facet strips and plot titles;
- 1.54 in below the panel (7.5 - 1.85 - 4.11), or 1.49 in inside the 0.05 in slide edge, for the x-axis chrome of `theme_hv_ppt_dark()`, `theme_hv_ppt_light()` and `theme_hv_ppt_slate()`: their 32 pt x-axis text and 40 pt x-axis title with its margin need about 1.48 in in Arial Bold, so a plot in these themes fits at full size.

The standard (4:3) rectangle is **provisional**: no 4:3 template is bundled, and the rectangle has not yet been checked on a real 4:3 deck. It follows the same rules, assuming the 4:3 title placeholder also ends near 1.33 in: the same top (1.85 in) and height (4.11 in), 2.3 in on the left for y labels and 0.45 in on the right, so 7.25 by 4.11 in at 2.3 in left.

In hvtiPlotR 2.8.14 and 2.8.15 both rectangles were 4.15 in high. Through hvtiPlotR 2.8.13 the rectangle was 8.79 by 3.9 in at 2.67 in left, 1.6 in top. Pass `panel_box = list(width = 8.79, height = 3.9, left = 2.67, top = 1.6)` to `save_ppt()` or `hv_add_plot()` to rebuild an old deck at its old geometry.

Value

A named list width, height, left, top, in inches, ready for the `panel_box` argument of `save_ppt()` and `hv_add_plot()`.

See Also

`save_ppt()`, `hv_add_plot()`, `hv_fit_location()`, `hv_ph_location()`

Examples

```

hv_panel_box()
hv_panel_box(format = "standard")

```

hv_panel_size	<i>Figure dimensions for a faceted plot</i>
---------------	---

Description

Chooses `ggsave()` dimensions from a plot's facet layout, so a faceted figure keeps panels of a readable size however many there are. A plot with one panel gets the standard 6 by 4 inch figure. A plot with several gets 3 by 2 inch panels, half the standard figure in each direction, plus the space its axes, titles, strips and legend need, measured by `hv_ggsave_dims()`.

Usage

```
hv_panel_size(
  plot,
  max_width = NULL,
  max_height = NULL,
  units = c("in", "cm", "mm")
)
```

Arguments

plot	A ggplot object. Faceted with <code>ggplot2::facet_wrap()</code> or <code>ggplot2::facet_grid()</code> , or not faceted at all. A patchwork is refused: its panels sit in nested tables the function cannot count.
max_width	The widest figure allowed, in units. NULL, the default, is 6.5 inches, a journal's text width, converted to units.
max_height	The tallest figure allowed, in units. NULL, the default, is 9 inches, converted to units; with the 6.5 inch width it fits a page.
units	One of "in", "cm", "mm". Default "in".

Details

When those panels, with the gutters and interior strips between them, would make the figure wider than `max_width` or taller than `max_height`, every panel shrinks by the same factor, so each keeps its 3:2 shape. A panel narrower than 1.6 inches leaves no room for 12 pt tick labels, so below that the function stops: the facets do not fit on one page, and `ggforce::facet_wrap_paginate()` splits them across several. A paginated plot is sized by the one page it draws.

The panel block is measured, not the whole figure, because the chrome around it varies: a long axis title or a legend at the top would otherwise come out of every panel.

Value

A named list with elements `width`, `height` and `units`, shaped to splat into `ggplot2::ggsave()` via `do.call()`, as `hv_ggsave_dims()` returns.

See Also

[hv_ggsave_dims\(\)](#), which measures the chrome.

Examples

```
p <- ggplot2::ggplot(mtcars, ggplot2::aes(hp, mpg)) +
  ggplot2::geom_point() +
  ggplot2::facet_wrap(~cyl)
dims <- hv_panel_size(p)
do.call(
  ggplot2::ggsave,
  c(list(filename = file.path(tempdir(), "facets.pdf"), plot = p), dims)
)
```

hv_ph_location	<i>Compute officer::ph_location() args for a fixed-panel slide layout</i>
----------------	---

Description

Given a fitted ggplot and a target panel rectangle (width/height/left/top on the slide), returns the width, height, left, top values to pass to [officer::ph_location\(\)](#) such that the *panel content area* of the plot lands at the specified slide coordinates, regardless of how much room the axis labels, axis titles, legend, plot title, or plot margins consume.

Usage

```
hv_ph_location(
  plot,
  panel_width,
  panel_height,
  panel_left,
  panel_top,
  units = c("in", "cm", "mm")
)
```

Arguments

plot	A ggplot (or patchwork) object.
panel_width	Target panel width, in units.
panel_height	Target panel height, in units.
panel_left	Distance from left edge of slide to the left edge of the panel, in units.
panel_top	Distance from top edge of slide to the top edge of the panel, in units.
units	One of "in", "cm", "mm". Default "in".

Details

The panel content area is the rectangular bounding box of the gtable cells tagged panel. Strip grobs that sit inside that bounding box are part of the target; strips outside it (e.g. facet_grid side strips, the strip row above a single-row facet_wrap) are treated as chrome.

Use with `save_ppt()` via `panel_box = list(width, height, left, top)` to apply this to every slide in a deck.

For a patchwork, the whole composition is measured, through `patchwork::patchworkGrob()`: the target rectangle holds the union of every plot's panels, with the axes and strips between them, and the outermost axes plus the `patchwork::plot_annotation()` title, subtitle and caption count as chrome. A nested patchwork counts as one panel, inner chrome included.

Value

A named list with elements width, height, left, top, all in units. Splat into `officer::ph_location()` via `do.call()`.

See Also

`hv_ggsave_dims()` for the sizing-only analogue used with `ggplot2::ggsave()`, and `save_ppt()` which accepts a `panel_box` argument that delegates to this helper.

Examples

```
p <- ggplot2::ggplot(mtcars, ggplot2::aes(hp, mpg)) +
  ggplot2::geom_point() +
  theme_hv_ppt_dark()

loc <- hv_ph_location(
  p,
  panel_width = 10, panel_height = 5,
  panel_left = 0.5, panel_top = 1.5
)
# doc <- officer::read_pptx(template)
# doc <- officer::add_slide(doc, layout = "Title and Content")
# doc <- officer::ph_with(
#   doc, rvg::dml(ggobj = p),
#   location = do.call(officer::ph_location, loc)
# )
```

Description

Every template bundled with hvtiPlotR (dark, light and slate) has the same six layouts by the same names, so a deck script runs on any of them. Name the role and pass the result to `officer::add_slide()`. `save_ppt()` sets a slide title, so its layout must have a title placeholder: it works with "figure", "section", "content" and "two". "title" (Title Slide has ctrTitle and subTitle placeholders) and "blank" (no placeholders) are for `officer::add_slide()` only.

Usage

```
hv_ppt_layout(
  role = c("figure", "section", "title", "content", "two", "blank")
)
```

Arguments

role	One of "figure" (title only, for a plot), "section" (a chapter head), "title" (the opening slide), "content" (title and one body), "two" (title and two bodies) or "blank".
------	---

Details

Through hvtiPlotR 2.8.11 the light template named its figure layout "Custom Layout" and its section layout "1_Custom Layout", and the dark template named its section layout "1_Title Only". `save_ppt()` still accepts those names, with a one-time message; calls to `officer::add_slide()` must use the new names.

Value

A single character string: the layout's name in every bundled template.

See Also

`save_ppt()`, `hv_panel_box()`.

Examples

```
hv_ppt_layout("figure")
hv_ppt_layout("section")
```

hv_ppt_palette

Series colors for a slide or a manuscript figure

Description

The colors `hv_ppt_series()` assigns, handed back as a plain character vector. This is the single definition of the series colors, so reach for it when a figure needs them outside the decorator rather than pasting hex codes into a script, where they drift the first time the palette changes.

Usage

```
hv_ppt_palette(mode = c("dark", "light"), n = NULL)
```

Arguments

mode	Slide or figure background. "dark" pairs with <code>theme_hv_ppt_dark()</code> , "light" with <code>theme_hv_ppt_light()</code> and <code>theme_hv_manuscript()</code> .
n	Number of colors to return, or NULL (default) for all six.

Details

The colors are the Okabe-Ito colorblind-safe palette, reordered for the background: high-luminance hues first on a dark slide, darker ones first on a light slide. Each ordering leads with its highest-contrast hue, 15.9:1 on a black panel and 5.2:1 on a white slide, but past that first entry the order is not a contrast ranking and should not be read as one. Black appears only in the light ordering, and it sits **last** there despite carrying the highest ratio of anything here (21:1 on white): house style draws annotation in the theme's ink, so a black series would be confusable with the label naming it. On a dark panel black is invisible, which is why that ordering omits it. These are not CORR brand colors.

Every dark color clears 3:1 against the black panel `theme_hv_ppt_dark()` draws, and every light color clears 3:1 on a white slide background, which is the contrast WCAG 2.2 (success criterion 1.4.11) asks of a graphical object a reader needs to read the chart. `theme_hv_ppt_light()` leaves its panel transparent so the slide shows through, so the light guarantee holds on a white template, which is what the house light template is. Check the colors again before using them on a tinted slide. One color in the light ordering is not Okabe-Ito's own. Their orange, #E69F00, reaches only 2.3:1 on white, so the light ordering uses #9D952B in its place, Okabe-Ito yellow darkened at the same hue until it clears 3:1. An orange darkened that far would sit too close to vermilion for a reader with red-green color-vision deficiency.

Six colors are supplied. Ask for more than that and you get an error rather than a silently recycled palette, because two series sharing a color is a worse outcome than a stopped script.

Value

A character vector of hex colors, in series order.

See Also

`hv_ppt_series()`, which applies these to a plot. Annotation text is drawn in the theme's ink, matching the axis, rather than in a series color, so these are not the values to label a curve with.

Examples

```
hv_ppt_palette("dark")
hv_ppt_palette("light", n = 4)
```

Description

You already know the two-step shape of a hvtiPlotR figure: a constructor prepares and validates the data, `plot()` hands back a bare `ggplot`, and you finish it with `+`. Across a deck that finishing step is the same few lines on every slide, and copying them plot by plot is how one figure quietly drifts out of step with the rest.

`hv_ppt_series()` bundles the finishing step into a single object you add once per plot. It carries the PowerPoint theme, a color scale and a matching shape scale, so a whole deck takes its look from one definition.

Usage

```
hv_ppt_series(
  mode = c("dark", "light"),
  colours = NULL,
  shapes = NULL,
  name = NULL,
  ...,
  colors = colours
)
```

Arguments

mode	Slide background the plot will sit on. "dark" pairs with <code>theme_hv_ppt_dark()</code> , "light" with <code>theme_hv_ppt_light()</code> .
colours	The same as colors: an alias kept for existing code. Give one or the other; both with different values is an error.
shapes	Numeric vector of <code>ggplot2</code> point shapes, one per group in order. Default NULL uses solid glyphs first.
name	Title used by both scales, so a drawn legend merges into one. Default NULL draws no title. Unused while the legend is off.
...	Additional named arguments forwarded to the underlying <code>theme_hv_ppt_*</code> () call, e.g. <code>base_size = 28</code> or <code>legend.position = "top"</code> .
colors	Character vector of colors, one per group in order. Default NULL uses the Okabe-Ito ordering for mode.

Details

The return value is a plain list, and `ggplot2`'s `+` unrolls a list element by element, so it composes like a theme:

```
ppt <- hv_ppt_series()
p1  <- p1 + ppt
p2  <- p2 + ppt
```

A theme cannot do this job alone. `ggplot2::theme()` governs the non-data ink, the text, panel, grid and ticks, and nothing that draws from the data, so no theme element sets a series color. Nor can a layer be smuggled in through the theme's `...`, which is forwarded to `ggplot2::theme()` and errors on anything that is not a theme element. Series color and shape live in the scales, and those are what this function adds alongside the theme.

Color and shape carry the same variable:

Both scales map the grouping column, so each series is told apart twice. That redundancy is the point. A projector in a dark room flattens color differences that read cleanly on your monitor, and a black and white handout drops them entirely, but the shapes survive both. The two scales share name, so on the rare draft that does draw a legend, `ggplot2` merges them into one instead of stacking two.

The default colors are the Okabe-Ito colorblind-safe palette, ordered for the background: high-luminance hues first on a dark slide, darker ones first on a light slide. On a light slide a darkened yellow stands in for Okabe-Ito orange, which is too faint against white; `hv_ppt_palette()` gives the reasoning. These are not CORR brand colors. Pass colors when a deck calls for a specific set.

Six colors and six shapes are supplied, which covers a grouping variable of up to six levels. A discrete scale errors when it runs out of values, so pass longer vectors for a wider variable.

No legend, by house standard:

Every `theme_hv_*` sets `legend.position = "none"`, and this decorator leaves that alone. CORR figures name the series where the series sits, on a slide, on a poster and in a manuscript alike, so the reader's eye never travels out to a key and back.

Annotation is drawn in the theme's ink, not in the color of the series it names: white on a dark slide, black on a light slide and in a manuscript. That is the ink the theme already gives the axis text and axis lines, so one ink for every label makes the annotation read as part of the figure's frame rather than as one more series.

```
plot(trends) +
  hv_ppt_series("dark") +
  ggplot2::annotate("text", x = 2015, y = 60, label = "Group I",
                    color = "white", size = 8)
```

The shape scale still earns its place with no legend drawn. Where two curves cross, the plotted summary points are what tells them apart.

Pass `legend.position = "top"` through `...` when a working draft wants a quick key, or hand the finished plot to `hv_legend_inside()` to drop one into an empty corner of the panel.

Grouped plots only:

The scales take effect on a plot that maps color to a grouping column, which means a constructor called with `group_col`. On an ungrouped plot they sit unused and only the theme applies. There is nothing to fix in that case; a single series needs no palette. To color the raw observations on an ungrouped plot, add your own layer after the decorator, for example `ggplot2::geom_point(color = "yellow")`.

Value

A list of ggplot2 components: a theme, a color scale and a shape scale. Add it to a ggplot object with +.

See Also

[hv_ppt_palette\(\)](#) for the colors themselves; [theme_hv_ppt_dark\(\)](#) and [theme_hv_ppt_light\(\)](#) for the themes this wraps; [hv_legend_inside\(\)](#) when a draft does want a key; [save_ppt\(\)](#) to export the finished plots to editable PowerPoint slides.

Examples

```
dta <- sample_trends_data(n = 400, seed = 42)
trends <- hv_trends(dta, x_col = "year", y_col = "value",
                    group_col = "group")

# One decorator, added to every plot in the deck.
ppt <- hv_ppt_series(mode = "dark")
plot(trends) + ppt

# House style carries no legend, so name the series where it sits. The
# label takes the same ink as the axis: white on a dark slide, black on a
# light one.
plot(trends) + ppt +
  ggplot2::annotate("text", x = 2015, y = 60, label = "Group I",
                    color = "white", size = 8)

# A light template, larger type, and your own colors.
plot(trends) +
  hv_ppt_series(
    mode      = "light",
    colors    = c("#0072B2", "#D55E00", "#009E73", "#CC79A7"),
    base_size = 24
  )
```

 hv_rmst_contrast

 Prepare RMST contrast data for plotting

Description

Validates a table of restricted mean survival time (RMST) differences, one row per weighting estimator, and returns an hv_rmst_contrast object. Call [plot.hv_rmst_contrast\(\)](#) on the result for a bare dot-and-whisker ggplot2 object: one point per estimator, a whisker for its interval, and a vertical reference line at zero.

Usage

```

hv_rmst_contrast(
  data,
  estimator = "estimator",
  diff = "diff_days",
  lo = "lo_days",
  hi = "hi_days",
  ...
)

```

Arguments

<code>data</code>	A data frame with one row per estimator, or a <code>ps_rmst</code> object (any list whose <code>\$tables\$estimates</code> is a data frame).
<code>estimator</code>	Name of the column holding estimator labels. Default <code>"estimator"</code> . If it is a factor, its levels set the display order; otherwise the order of first appearance is used.
<code>diff</code>	Name of the numeric column with the RMST difference (in days). Default <code>"diff_days"</code> .
<code>lo, hi</code>	Names of the numeric columns with the lower and upper interval limits (in days). Defaults <code>"lo_days"</code> and <code>"hi_days"</code> .
<code>...</code>	Ignored; present for S3 consistency.

Details

The input follows the `tables$estimates` contract of `hvtiRpropensity::ps_rmst()`: columns `estimator`, `subset`, `weighting`, `n`, `ess_treated`, `ess_control`, `rmst_treated`, `rmst_control`, `diff`, `diff_days`, `lo_days`, `hi_days` and `n_failed`, where a positive `diff` favors the treated arm. Only the estimator, difference and interval columns are used here. `hvtiRpropensity` is not required: pass a plain data frame, or the `ps_rmst` object itself and its `$tables$estimates` is read.

Value

An object of class `c("hv_rmst_contrast", "hv_data")`: a list with

- `$data` The estimates data frame, with the estimator column as supplied.
- `$meta` Named list: `estimator`, `diff`, `lo`, `hi` (column names), `estimator_levels`, `n_estimators`, `n_missing`.
- `$tables` Empty list.

See Also

[plot.hv_rmst_contrast\(\)](#), [hv_rmst_curves\(\)](#)

Other Propensity Score & Matching: [hv_mirror_hist\(\)](#), [hv_rmst_curves\(\)](#), [plot.hv_mirror_hist\(\)](#), [plot.hv_rmst_contrast\(\)](#), [plot.hv_rmst_curves\(\)](#), [sample_covariate_balance_data\(\)](#)

Examples

```
est <- data.frame(
  estimator = factor(c("IPTW", "Overlap", "Matching"),
    levels = c("IPTW", "Overlap", "Matching")),
  diff_days = c(41, 35, 28),
  lo_days   = c(-5, 2, -14),
  hi_days   = c(87, 68, 70)
)
rc <- hv_rmst_contrast(est)
rc

plot(rc) +
  ggplot2::labs(y = NULL) +
  theme_hv_manuscript()
```

 hv_rmst_curves

Prepare weighted Kaplan-Meier curves for an RMST figure

Description

Validates the step-function curves behind an RMST analysis and returns an `hv_rmst_curves` object. Call `plot.hv_rmst_curves()` for a bare `ggplot2` object with one facet per estimator and one step curve per arm. With `tau`, the area under each arm's curve up to `tau` is shaded (that area is the RMST) and a vertical line marks `tau`. With `estimates`, each facet is annotated with its RMST difference and interval.

Usage

```
hv_rmst_curves(
  curves,
  estimates = NULL,
  tau = NULL,
  facet = "estimator",
  arm = "arm",
  time = "time",
  surv = "surv",
  ...
)
```

Arguments

<code>curves</code>	A data frame of curve steps, or a <code>ps_rmst</code> object.
<code>estimates</code>	Optional data frame with the <code>hv_rmst_contrast()</code> columns (<code>estimator</code> , <code>diff_days</code> , <code>lo_days</code> , <code>hi_days</code>), used for the facet annotations. If <code>NULL</code> and <code>curves</code> is a <code>ps_rmst</code> -like object, its <code>\$tables\$estimates</code> is used when present. <code>NULL</code> otherwise: no annotation.

tau	Optional single positive number: the RMST horizon, on the same scale as time.
facet	Name of the column defining facets. Default "estimator". Factor levels set the facet order.
arm	Name of the arm column. Default "arm". A character column is ordered "treated", "control", then any other value.
time, surv	Names of the time and survival columns. Defaults "time" and "surv".
...	Ignored; present for S3 consistency.

Details

The input follows the `tables$curves` contract of `hvtiRpropensity::ps_rmst()`: columns `estimator`, `arm` ("treated" or "control"), `time` and `surv`, as right-continuous steps that start at time 0 with `surv` 1. `hvtiRpropensity` is not required: pass plain data frames, or the `ps_rmst` object itself as `curves` and its `$tables$curves` (and, when `estimates` is `NULL`, `$tables$estimates`) are read.

Value

An object of class `c("hv_rmst_curves", "hv_data")`: a list with

`$data` The curves data frame.

`$meta` Named list: `facet`, `arm`, `time`, `surv` (column names), `tau`, `facet_levels`, `arm_levels`, `n_missing`.

`$tables` `shade`: the staircase polygon under each curve up to `tau` (zero rows when `tau` is `NULL`); `labels`: one annotation per facet (zero rows when `estimates` is `NULL`).

See Also

[plot.hv_rmst_curves\(\)](#), [hv_rmst_contrast\(\)](#)

Other Propensity Score & Matching: [hv_mirror_hist\(\)](#), [hv_rmst_contrast\(\)](#), [plot.hv_mirror_hist\(\)](#), [plot.hv_rmst_contrast\(\)](#), [plot.hv_rmst_curves\(\)](#), [sample_covariate_balance_data\(\)](#)

Examples

```
steps <- function(est, arm, rate) {
  t <- c(0, sort(round(stats::rexp(40, rate) * 365)))
  data.frame(estimator = est, arm = arm, time = t,
             surv = c(1, 1 - seq_len(40) / 41))
}
set.seed(1)
curves <- rbind(
  steps("IPTW", "treated", 1 / 3), steps("IPTW", "control", 1 / 2),
  steps("Overlap", "treated", 1 / 3), steps("Overlap", "control", 1 / 2)
)
est <- data.frame(estimator = c("IPTW", "Overlap"),
                  diff_days = c(41, 35), lo_days = c(-5, 2), hi_days = c(87, 68))

rc <- hv_rmst_curves(curves, estimates = est, tau = 365)
rc
```

```
plot(rc) +
  ggplot2::labs(x = "Days", y = "Survival") +
  theme_hv_manuscript()
```

hv_role_palette

Colors for a figure's levels, by role

Description

The house color rule for EDA and follow-up figures, as a named vector you can check or reuse: an event is vermillion, censored is blue, missing is light gray, and every other level takes a colorblind-safe color in order. To apply the rule to a plot, use [scale_fill_hv\(\)](#) or [scale_color_hv\(\)](#), which call this for each panel.

Usage

```
hv_role_palette(levels, event = NULL, censored = NULL, missing = "(Missing)")
```

Arguments

levels	Character vector of distinct level names, in drawing order.
event, censored	Level names that take the event or the censored color, or NULL for none. Names not in levels are ignored.
missing	Level name that stands for missing values. "(Missing)" is the explicit level hv_eda() adds.

Details

Every color is decided from levels alone:

1. A level named in event is #D55E00, in censored #0072B2, and in missing #CCCCCC. These hold even when the level is the only one.
2. Every other level, in levels order, takes the light series of [hv_ppt_palette\(\)](#): Okabe-Ito blue, vermillion, bluish green and reddish purple, then a darkened yellow and black, so a sole level with no role is blue.
3. When any of levels is named in event or censored, blue and vermillion leave that series, so no other level can be mistaken for the event or for censored. A role name absent from levels changes nothing, so one call can serve panels that do and do not carry the event.
4. When the other levels outnumber the series, all of them switch to Paul Tol's muted palette, without its sand and cyan, and without its rose when a role is present. Beyond that there are no more colorblind-safe colors, and the function stops with an error of class hv_role_palette_capacity.

Every color except the missing gray is at least 2:1 against white. The gray is fainter on purpose: missing is the one level meant to recede.

Value

A character vector of hex colors, named by levels and in the same order.

See Also

[scale_fill_hv\(\)](#) and [scale_color_hv\(\)](#) to apply the rule to a plot; [hv_ppt_palette\(\)](#) for series colors with no role attached.

Examples

```
hv_role_palette(c("No event", "Reoperation", "Death"),
               event = "Death", censored = "No event")
hv_role_palette(c("Female", "Male", "(Missing)"))
```

hv_sankey

Prepare cluster stability Sankey data for plotting

Description

Validates a wide cluster-assignment data frame, resolves node level ordering, computes default node colors if not supplied, and pre-computes the long-format Sankey data. Call [plot.hv_sankey](#) on the result to obtain a bare ggplot2 Sankey diagram using **ggsankey** geoms.

Usage

```
hv_sankey(
  data,
  cluster_cols = paste0("C", 2:9),
  node_levels = NULL,
  node_colours = NULL,
  node_colors = node_colours
)
```

Arguments

data	Data frame; one row per patient. Must contain all columns named in cluster_cols.
cluster_cols	Character vector of column names giving the cluster assignments at each value of K, in ascending K order. Default paste0("C", 2:9).
node_levels	Character vector giving the display order of node labels (bottom to top within each column). If NULL (default), a lineage-preserving order is derived from the data so each child cluster sits next to its parent and flows stay uncrossed (see Details). If supplied, it is used verbatim but must cover every observed cluster label.
node_colours	The same as node_colors: an alias kept for existing code. Give one or the other; both with different values is an error.

node_colors Named character vector mapping node labels to fill colors. If NULL (default), labels take Paul Tol's muted palette, nine colorblind-safe colors, in `node_levels` order. When there are more labels than palette colors the palette is recycled with a warning. A named vector that covers only some labels overrides those and leaves the rest at their default color, so `c(G = "#CC79A7")` recolors one node. An unnamed vector is used as given.

Details

By default (`node_levels = NULL`) the node order is read from the data, not from one column's factor levels. We take every label that appears in any `cluster_cols` column, so no finer-K cluster is dropped to NA, and place each child next to its parent: the coarser-K cluster that contributes most of its members. Siblings end up together, the way leaves sit in a dendrogram, and the flows stay short and uncrossed. Pass `node_levels` yourself to override this; it is then used as given, but it must name every label in the data.

Value

An object of class `c("hv_sankey", "hv_data")`:

`$data` The long-format Sankey data frame (four columns: `x`, `node`, `next_x`, `next_node`).

`$meta` Named list: `cluster_cols`, `node_levels`, `node_colors` (also as `node_colours`, for existing code), `n_patients`, `n_k`.

`$tables` Empty list.

See Also

Worked recipe with rendered output: <https://ehrlinger.github.io/hvtiGraphics/sankey.html>.

[plot.hv_sankey](#), [sample_cluster_sankey_data](#)

Examples

```
dta <- sample_cluster_sankey_data(n = 300, seed = 42)

if (requireNamespace("ggsankey", quietly = TRUE)) {
  # 1. Build data object
  sn <- hv_sankey(dta)
  sn # prints cluster cols and node count

  # 2. Bare plot -- undecorated ggplot returned by plot.hv_sankey
  p <- plot(sn)

  # 3. Decorate: axis labels and theme
  p +
    ggplot2::labs(x = NULL, title = "Cluster Stability: K = 2 to 9") +
    theme_hv_poster()
}
```

 hv_spaghetti

 Prepare spaghetti / profile data for plotting

Description

Validates a long-format repeated-measures data frame and returns an `hv_spaghetti` object. Call `plot.hv_spaghetti` on the result to obtain a bare `ggplot2` trajectory plot that you can decorate with color scales, axis labels, and `theme_hv_manuscript`.

Usage

```
hv_spaghetti(
  data,
  x_col = "time",
  y_col = "value",
  id_col = "id",
  colour_col = NULL,
  color_col = colour_col
)
```

Arguments

<code>data</code>	Data frame; one row per observation per subject.
<code>x_col</code>	Name of the time column. Default "time".
<code>y_col</code>	Name of the outcome column. Default "value".
<code>id_col</code>	Name of the subject-identifier column (used as the group aesthetic for line continuity). Default "id".
<code>colour_col</code>	The same as <code>color_col</code> : an alias kept for existing code. Give one or the other; both with different values is an error.
<code>color_col</code>	Name of the column to map to line color, or NULL for a single uniform color. Default NULL.

Value

An object of class `c("hv_spaghetti", "hv_data")`; call `plot()` on the result to render the figure; see `plot.hv_spaghetti`. The list contains:

`$data` The validated input data frame.

`$meta` Named list: `x_col`, `y_col`, `id_col`, `color_col` (also as `colour_col`, for existing code), `n_subjects`, `n_obs`.

`$tables` Empty list.

See Also

Worked recipe with rendered output: <https://ehrlinger.github.io/hvtiGraphics/spaghetti.html>.

`plot.hv_spaghetti` to render as a ggplot2 figure, `theme_hv_manuscript` for the publication theme, `sample_spaghetti_data` for example data.

Other Spaghetti plot: `plot.hv_spaghetti()`

Examples

```
dta <- sample_spaghetti_data(n_patients = 150, seed = 42)

# 1. Build data object
sp <- hv_spaghetti(dta, color_col = "group")
sp # prints subject count, observation count, column mapping

# 2. Bare plot -- undecorated ggplot returned by plot.hv_spaghetti
p <- plot(sp)

# 3. Decorate: color palette, axis labels, theme
p +
  ggplot2::scale_color_manual(
    values = c(Female = "firebrick", Male = "steelblue"), name = NULL
  ) +
  ggplot2::labs(x = "Years after Operation",
                y = "AV Mean Gradient (mmHg)") +
  theme_hv_poster()
```

hv_stacked

Prepare stacked histogram data for plotting

Description

Validates a patient-level or observation-level data frame and returns an `hv_stacked` object. Call `plot.hv_stacked` on the result to obtain a bare ggplot2 stacked (or proportional) histogram that you can decorate with color scales, axis labels, and `theme_hv_manuscript`.

Usage

```
hv_stacked(
  data,
  x_col = "year",
  group_col = "category",
  binwidth = 1,
  position = c("stack", "fill")
)
```

Arguments

<code>data</code>	A data frame.
<code>x_col</code>	Name of the numeric column to bin along the x-axis. Default "year".
<code>group_col</code>	Name of the column whose distinct values define the stacked groups. Will be coerced to a factor inside the aesthetic mapping. Default "category".
<code>binwidth</code>	Width of each histogram bin, in the same units as <code>x_col</code> . Default 1.
<code>position</code>	Bar position: "stack" (raw counts, the default) or "fill" (proportions that sum to 1 within each bin).

Value

An object of class `c("hv_stacked", "hv_data")`:

`$data` The validated input data frame.

`$meta` Named list: `x_col`, `group_col`, `binwidth`, `position`, `n_obs`, `n_groups`.

`$tables` Empty list.

See Also

Worked recipe with rendered output: <https://ehrlinger.github.io/hvtiGraphics/histograms.html>.

`plot.hv_stacked`, `sample_stacked_histogram_data`

Examples

```
dta <- sample_stacked_histogram_data()

# 1. Build data object
sh <- hv_stacked(dta, x_col = "year", group_col = "category")
sh # prints obs / group count

# 2. Bare plot -- undecorated ggplot returned by plot.hv_stacked
p <- plot(sh)

# 3. Decorate: fill/color brewer palette, axis labels, theme
p +
  ggplot2::scale_fill_manual(values = hv_palette(), name = "Category") +
  ggplot2::scale_color_manual(values = hv_palette(), name = "Category") +
  ggplot2::labs(x = "Year", y = "Count") +
  theme_hv_poster()
```

hv_survival	<i>Prepare survival data for plotting</i>
-------------	---

Description

Fits a Kaplan-Meier (product-limit) or Nelson-Aalen (Fleming-Harrington) survival model to patient-level data and returns an `hv_survival` object containing the tidy model output and accessory tables. No plot is built at this stage; call `plot.hv_survival` on the result to obtain a bare `ggplot2` object that you can decorate with scales, labels, and `theme_hv_manuscript`.

Usage

```
hv_survival(
  data,
  time_col = "iv_dead",
  event_col = "dead",
  group_col = NULL,
  method = c("kaplan-meier", "nelson-aalen"),
  conf_level = 0.95,
  report_times = c(1, 5, 10, 15, 20, 25)
)
```

Arguments

<code>data</code>	A data frame with one row per patient.
<code>time_col</code>	Name of the numeric column holding follow-up time (in years). Default "iv_dead".
<code>event_col</code>	Name of the 0/1 or logical event-indicator column. Default "dead".
<code>group_col</code>	Optional name of a character or factor column used to stratify the analysis. NULL (default) produces an unstratified estimate labeled "All".
<code>method</code>	Estimator: "kaplan-meier" (default, logit CI; mirrors SAS %kaplan) or "nelson-aalen" (Fleming-Harrington cumulative hazard with log CI; mirrors SAS %nelsont, preferred when $S(t)$ approaches zero).
<code>conf_level</code>	Confidence level for the CI band. Default 0.95.
<code>report_times</code>	Numeric vector of time points at which survival estimates and numbers-at-risk are tabulated. Default c(1, 5, 10, 15, 20, 25).

Value

An object of class `c("hv_survival", "hv_data")` (a list); call `plot()` on the result to render the figure; see `plot.hv_survival`. The list has three elements:

`$data` Tidy data frame with one row per (time, strata) pair. Columns: time, surv, lower, upper, n.risk, n.event, n.censor, cumhaz, strata, hazard, density, mid_time, life, proplife, log_cumhaz, log_time.

`$meta` Named list: time_col, event_col, group_col, method, conf_level, report_times, n_obs (the *analyzed* cohort), n_input, n_excluded, n_events.

`$tables` Named list with two data frames: `risk` (`strata`, `report_time`, `n.risk`), where `n.risk` counts analyzed subjects with follow-up greater than or equal to the exact report time, and `report` (`strata`, `report_time`, `surv`, `lower`, `upper`, `n.risk`, `n.event`).

Missing data

Rows with a missing time, event, or grouping value are excluded before fitting, with a warning naming the columns responsible. `$meta$n_obs` and `print()` report the *analyzed* cohort, alongside `n_input` and `n_excluded`, so the reported N always matches the cohort the estimates describe.

References

SAS templates: `tp.ac.dead.sas (%kaplan, %nelson)`.

See Also

Worked recipe with rendered output: <https://ehrlinger.github.io/hvtiGraphics/survival.html>.

`plot.hv_survival` to render as a ggplot2 figure, `theme_hv_manuscript` for the publication theme, `sample_survival_data` for example data.

Other Kaplan-Meier survival: `plot.hv_survival()`, `sample_survival_data()`

Examples

```
dta <- sample_survival_data(n = 500, seed = 42)

# 1. Build data object
km <- hv_survival(dta)
km          # print method shows key metadata
km$tables$report # survival estimates at report_times
km$tables$risk   # numbers at risk

# 2. Bare plot -- undecorated ggplot returned by plot.hv_survival
p <- plot(km)

# 3. Decorate: axis scales, labels, theme
p +
  ggplot2::scale_y_continuous(breaks = seq(0, 100, 20),
                              labels = function(x) paste0(x, "%")) +
  ggplot2::scale_x_continuous(breaks = seq(0, 20, 5)) +
  ggplot2::coord_cartesian(xlim = c(0, 20), ylim = c(0, 100)) +
  ggplot2::labs(x = "Years after Operation", y = "Survival (%)",
                title = "Freedom from Death") +
  theme_hv_poster()

# Stratified: color scale adds clinical meaning
dta_s <- sample_survival_data(
  n = 500, strata_levels = c("Type A", "Type B"),
  hazard_ratios = c(1, 1.4), seed = 42
)
km_s <- hv_survival(dta_s, group_col = "valve_type")
```

```

plot(km_s) +
  ggplot2::scale_color_manual(
    values = c("Type A" = "steelblue", "Type B" = "firebrick"),
    name = "Valve Type"
  ) +
  ggplot2::labs(x = "Years after Operation", y = "Survival (%)") +
  theme_hv_poster()

# Other plot types
plot(km, type = "cumhaz") +
  ggplot2::labs(x = "Years", y = "Cumulative Hazard") +
  theme_hv_poster()

plot(km, type = "loglog") +
  ggplot2::labs(x = "log(Years)", y = "log(-log S(t))",
    title = "PH Assumption Check") +
  theme_hv_poster()

# --- Global theme + RColorBrewer (set once per session) -----

old <- ggplot2::theme_set(theme_hv_manuscript())
plot(km_s) +
  ggplot2::scale_color_manual(values = hv_palette(), name = "Valve Type") +
  ggplot2::labs(x = "Years after Operation", y = "Survival (%)")
ggplot2::theme_set(old)

# See vignette("plot-decorators", package = "hvtiPlotR") for theming,
# color scales, annotation labels, and saving plots.

```

hv_survival_difference

Prepare survival difference (life-gained) data for plotting

Description

Stores pre-computed survival difference data as an `hv_survival_difference` object. Pass the result to `plot.hv_survival_difference()` to render the plot. Covers `tp.hp.dead.life-gained.sas` and the survival-difference component of `tp.hp.numtreat.survdiff.matched.sas`.

Usage

```

hv_survival_difference(
  diff_data,
  x_col = "time",
  estimate_col = "difference",
  lower_col = NULL,
  upper_col = NULL,
  group_col = NULL
)

```

Arguments

diff_data	Data frame of pre-computed survival differences. See sample_survival_difference_data() .
x_col	Name of the time column. Default "time".
estimate_col	Name of the difference column. Default "difference".
lower_col	Lower CI column, or NULL. Default NULL.
upper_col	Upper CI column, or NULL. Default NULL.
group_col	Grouping column for multiple comparisons, or NULL. Default NULL.

Value

An S3 object of class `c("hv_survival_difference", "hv_data")`.

See Also

Worked recipe with rendered output: <https://ehrlinger.github.io/hvtiGraphics/nnt.html>.
[plot.hv_survival_difference\(\)](#), [sample_survival_difference_data\(\)](#), [hv_nnt\(\)](#)

Examples

```
library(ggplot2)

diff_dat <- sample_survival_difference_data(
  groups = c("Control" = 1.0, "Treatment" = 0.70)
)

sd <- hv_survival_difference(diff_dat,
  lower_col = "diff_lower", upper_col = "diff_upper"
)

plot(sd) +
  geom_hline(yintercept = 0, linetype = "dashed", color = "gray50") +
  scale_x_continuous(limits = c(0, 10), breaks = 0:10) +
  scale_y_continuous(limits = c(-5, 30),
    labels = function(x) paste0(x, "%")) +
  labs(x = "Years", y = "Survival Difference (%)") +
  theme_hv_poster()
```

hv_trends

Prepare temporal trend data for plotting

Description

Validates a patient-level data frame, computes per-x-value summary statistics (mean or median), and returns an `hv_trends` object. Call [plot.hv_trends](#) on the result to obtain a bare `ggplot2` trend plot (LOESS smooth + annual summary points) that you can decorate with color scales, axis limits, and [theme_hv_manuscript](#).

Usage

```

hv_trends(
  data,
  x_col = "year",
  y_col = "value",
  group_col = "group",
  summary_fn = c("mean", "median")
)

```

Arguments

<code>data</code>	Patient-level data frame (one row per patient).
<code>x_col</code>	Name of the numeric/integer time column (e.g. surgery year). Default "year".
<code>y_col</code>	Name of the continuous outcome column. Default "value".
<code>group_col</code>	Name of the grouping column, or NULL for a single group. Default "group".
<code>summary_fn</code>	Function used to compute the per-x-point estimate: "mean" or "median". Default "mean".

Value

An object of class `c("hv_trends", "hv_data")`; call `plot()` on the result to render the figure (see [plot.hv_trends](#)). The list contains:

`$data` The original patient-level data frame.

`$meta` Named list: `x_col`, `y_col`, `group_col`, `summary_fn`, `n_obs`, `n_groups`.

`$tables` List with one element: `summary`, a data frame of per-x (per-group) summary statistics used for the point overlay.

See Also

Worked recipe with rendered output: <https://ehrlinger.github.io/hvtiGraphics/spaghetti.html>.

[plot.hv_trends](#) to render as a ggplot2 figure, [theme_hv_manuscript](#) for the publication theme, [sample_trends_data](#) for example data.

Other Temporal trends: [plot.hv_trends\(\)](#)

Examples

```

dta <- sample_trends_data(n = 600, year_range = c(1985L, 2015L),
  groups = c("I", "II", "III", "IV"))

# 1. Build data object
tr <- hv_trends(dta, summary_fn = "median")
tr # prints observation and group counts

# 2. Bare plot -- undecorated ggplot returned by plot.hv_trends
p <- plot(tr)

```

```
# 3. Decorate: color palette, axis scales, labels, theme
p +
  ggplot2::scale_color_manual(
    values = c(I = "steelblue", II = "firebrick",
              III = "forestgreen", IV = "goldenrod3"),
    name = "NYHA Class"
  ) +
  ggplot2::scale_x_continuous(limits = c(1985, 2015),
                             breaks = seq(1985, 2015, 5)) +
  ggplot2::scale_y_continuous(limits = c(0, 80),
                             breaks = seq(0, 80, 20)) +
  ggplot2::coord_cartesian(xlim = c(1985, 2015), ylim = c(0, 80)) +
  ggplot2::labs(x = "Years", y = "%") +
  theme_hv_poster()
```

hv_upset

Prepare UpSet co-occurrence data for plotting

Description

Validates a set-membership data frame, checks all intersect columns are binary (logical or 0/1 integer), computes per-set counts, and returns an hv_upset object. Call `plot.hv_upset` on the result to obtain a ggplot2 UpSet diagram (backed by `scale_x_upset`).

Usage

```
hv_upset(data, intersect)
```

Arguments

data	A data frame. Each set-membership column must be logical or integer (0/1).
intersect	Character vector of column names to treat as sets. Must contain at least two names that exist in data.

Value

An object of class `c("hv_upset", "hv_data")`:

`$data` The validated input data frame, plus a `.Procedures` list-column derived from the indicator matrix (consumed by `scale_x_upset()`).

`$meta` Named list: `intersect`, `n_patients`, `n_sets`.

`$tables` List with one element: `set_counts`, a named integer vector of per-set patient counts.

See Also

Worked recipe with rendered output: <https://ehrlinger.github.io/hvtiGraphics/upset.html>.

`plot.hv_upset`, `sample_upset_data`

Examples

```
sets <- c("AV_Replacement", "AV_Repair", "MV_Replacement", "MV_Repair",
          "TV_Repair", "Aorta", "CABG")
dta <- sample_upset_data(n = 300, seed = 42)

# 1. Build data object
up <- hv_upset(dta, intersect = sets)
up # prints set counts

# 2. Plot. The default (set_size = TRUE) returns a patchwork composite,
# so apply themes with `&` to theme every sub-panel. Use `+` for
# set_size = FALSE (a plain ggplot).
plot(up) & theme_hv_poster()
plot(up, set_size = FALSE) + theme_hv_poster()
```

 hv_venn

 Prepare a Venn diagram for plotting

Description

Validates a wide set-membership data frame and returns an `hv_venn` object carrying a region-count table. Call `plot.hv_venn` on the result for a bare **ggplot2** Venn diagram. `hv_venn()` is the small-set-count companion to `hv_upset`, reading the same input; for more than three sets use `hv_upset`.

Usage

```
hv_venn(data, sets)
```

Arguments

<code>data</code>	A data frame; one row per patient. Each set column must be logical or 0/1 numeric.
<code>sets</code>	Character vector of 2 to 3 column names to draw as sets.

Value

An object of class `c("hv_venn", "hv_data")`:

`$data` The input data frame.

`$meta` Named list: `sets`, `n_patients`, `n_sets`.

`$tables$regions` A data frame with one logical column per set (the in/out membership pattern), a region label, and `n` (patients in that exact region).

See Also

Worked recipe with rendered output: <https://ehrlinger.github.io/hvtiGraphics/upset.html>.

`plot.hv_venn`, `hv_upset`, `sample_upset_data`

Examples

```
dta <- sample_upset_data(n = 300, seed = 42)
v <- hv_venn(dta, sets = c("AV_Replacement", "MV_Replacement", "CABG"))
v$tables$regions
```

hvtiPlotR-themes	<i>hvtiPlotR ggplot2 themes</i>
------------------	---------------------------------

Description

Drop-in replacements for `ggplot2::theme_bw()` and friends, carrying the Cardiovascular Outcomes, Registries and Research (CORR) house style for four publication contexts:

Usage

```
theme_hv_manuscript(
  base_size = 12,
  base_family = "",
  header_family = NULL,
  base_line_size = base_size/22,
  base_rect_size = base_size/22,
  ink = "black",
  paper = "white",
  accent = "#3366FF",
  ...
)

theme_hv_poster(
  base_size = 16,
  base_family = "",
  header_family = NULL,
  base_line_size = base_size/22,
  base_rect_size = base_size/22,
  ink = "black",
  paper = "white",
  accent = "#3366FF",
  ...
)

theme_hv_ppt_dark(
  base_size = 32,
  base_family = "Arial",
  header_family = NULL,
  base_line_size = base_size/22,
  base_rect_size = base_size/22,
  ink = "white",
```

```
    paper = "transparent",
    accent = "#3366FF",
    ...
)

theme_hv_ppt_slate(
  base_size = 32,
  base_family = "Arial",
  header_family = NULL,
  base_line_size = base_size/22,
  base_rect_size = base_size/22,
  ink = "white",
  paper = "transparent",
  accent = "#3366FF",
  ...
)

theme_hv_ppt_light(
  base_size = 32,
  base_family = "Arial",
  header_family = NULL,
  base_line_size = base_size/22,
  base_rect_size = base_size/22,
  ink = "black",
  paper = "transparent",
  accent = "#3366FF",
  ...
)

hv_theme_manuscript(...)

theme_manuscript(...)

theme_man(...)

hv_theme_poster(...)

theme_poster(...)

hv_theme_dark_ppt(...)

theme_dark_ppt(...)

hv_theme_ppt(...)

theme_ppt(...)

hv_theme_light_ppt(...)
```

```
theme_light_ppt(...)
```

Arguments

<code>base_size</code>	Base font size in points.
<code>base_family</code>	Base font family. Default "" (device default).
<code>header_family</code>	Font family for headers, or NULL to inherit <code>base_family</code> . Default NULL.
<code>base_line_size</code>	Line size used for axis lines and borders. Default <code>base_size / 22</code> .
<code>base_rect_size</code>	Rectangle border size. Default <code>base_size / 22</code> .
<code>ink</code>	Foreground (text and line) color.
<code>paper</code>	Background color.
<code>accent</code>	Accent color used by some <code>theme_gray()</code> elements. Default "#3366FF".
<code>...</code>	Additional named theme elements forwarded to a final <code>ggplot2::theme()</code> call. Use this to override any theme element from the call site, e.g. <code>legend.position = "right"</code> .

Details

- `theme_hv_manuscript()`: clean white background for journal figures
- `theme_hv_poster()`: medium-font theme with visible axis lines for posters
- `theme_hv_ppt_dark()`: dark panel background, white text, large font
- `theme_hv_ppt_light()`: light/transparent panel background, black text

Each theme follows the `theme_bw()` contract: pass `base_size / base_family` to control global typography, then chain a `+ theme(...)` call to override anything else. Any extra named argument goes straight into a final `ggplot2::theme()` call; tweak elements at the call site without a separate `+ chain`:

```
theme_hv_manuscript(legend.position = "right")
theme_hv_ppt_dark(plot.title = element_text(hjust = 0.5))
```

Your elements override the hvtiPlotR defaults.

`theme_hv_ppt_dark()` is the default PPT theme: a black panel with white text suited to dark slide backgrounds. Use `theme_hv_ppt_light()` when the slide template is light. Both PPT themes hide the legend by default; override with `legend.position = "right"` (or `chain + theme(...)`).

Margins on axis text/title are scaled from `base_size` via the standard `half_line = base_size / 2` convention; a deck at `base_size = 28` and one at `base_size = 36` both feel proportionate without you touching the margins.

`base_family` defaults to "Arial", the CORR slide standard. Arial isn't one of R's built-in PostScript fonts, so a device that resolves fonts through that closed metrics table (`postscript()`/`pdf()`, e.g. R CMD check or headless Linux rendering) cannot draw it directly. When that happens, the plot falls back to "Helvetica" (metrically compatible) automatically at draw time, with a one-time session message explaining why; devices that resolve fonts via the OS (quartz, cairo, RStudio's

graphics device) are unaffected and keep using real Arial. No global font registry is modified. See `.hv_family_resolves()` in `ppt-font-fallback.R` for the mechanism.

`theme_hv_ppt_slate()` pairs with the slate template (`hv_ppt_template_slate.pptx`): the dark theme's type and axes, white on the slate background, with a white plotting panel. Color the data inside the panel from `hv_ppt_palette("light")`, and give text drawn inside the panel (`geom_text()`, `annotate()`) a dark color, since the theme's ink is white. Axis ticks point inward and are dark gray, so they show on the white panel. Geoms without their own color draw in black on the white panel.

Value

A `ggplot2::theme()` object.

See Also

[save_ppt\(\)](#) to export a themed plot to an editable PowerPoint slide; [make_footnote\(\)](#) to annotate a draft figure before publication.

Examples

```
p <- ggplot2::ggplot(mtcars, ggplot2::aes(wt, mpg)) +
  ggplot2::geom_point() +
  ggplot2::labs(x = "Weight (1000 lb)", y = "Miles per gallon")
p + theme_hv_manuscript()
p + theme_hv_poster()
p + theme_hv_ppt_dark()
p + theme_hv_ppt_light()
p + theme_hv_ppt_slate()

# Extra named arguments go straight into a final theme() call.
p + theme_hv_manuscript(base_size = 10, legend.position = "right")
```

is_hv_data

Test whether an object is an hvtiPlotR data object

Description

Test whether an object is an hvtiPlotR data object

Usage

```
is_hv_data(x)
```

Arguments

x Any R object.

Value

TRUE if x inherits from "hv_data", FALSE otherwise.

See Also

`summary.hv_data()`, `autoplot.hv_data()`, `as.data.frame.hv_data()`

Examples

```
d <- sample_correlation_data(n = 50)
is_hv_data(hv_correlation_matrix(d, c("a1c", "glucose")))
is_hv_data(d)
```

make_footnote

Add a Draft Footnote to a Figure

Description

Writes a small text annotation in the bottom-right corner of the **current** graphics device using grid. Call this *after* printing or displaying the plot. For publication-ready figures, omit the call; the plot is unchanged.

Usage

```
make_footnote(
  text = getwd(),
  timestamp = TRUE,
  prefix = "DRAFT - ",
  size = 0.7,
  colour = gray(0.5),
  x = 1,
  y = 0,
  hjust = "right",
  vjust = "bottom",
  margin_mm = 2,
  color = colour
)

makeFootnote(
  footnoteText = getwd(),
  size = 0.7,
  color = gray(0.5),
  timestamp = TRUE
)
```

Arguments

text	Text to display. Defaults to the current working directory; for most uses pass the script filename instead: <code>make_footnote("R/analysis.R")</code> .
timestamp	Logical; if TRUE, <code>Sys.time()</code> is appended to text. Default TRUE. Set to FALSE for reproducible output.
prefix	String prepended to text before the timestamp. Default "DRAFT \u2014 ". Set to "" to suppress the prefix.
size	Font size as a multiplier relative to the device default (passed to <code>grid::gpar()</code> as <code>cex</code>). Default 0.7.
colour	The same as <code>color</code> : an alias kept for existing code. Give one or the other; both with different values is an error.
x	Horizontal position in normalized parent coordinates ("npc"). Default 1 (right edge). Decrease to move left.
y	Vertical position in "npc". Default 0 (bottom). Increase to move up.
hjust	Horizontal justification: "right" (default), "left", or "center".
vjust	Vertical justification: "bottom" (default), "top", or "center".
margin_mm	Margin in mm pulled back from the x/y position. Default 2.
color	Font color. Default <code>gray(0.5)</code> (medium gray), which is visually unobtrusive on both screen and print.
footnoteText	Equivalent to text in <code>make_footnote()</code> .

Details**Typical workflow:**

```
# During analysis (draft)
p <- plot(hv_hazard(...)) + theme_hv_manuscript()
print(p)
make_footnote("analysis/mortality.R") # adds draft annotation

# For publication – just don't call make_footnote()
ggsave("figures/fig1.pdf", p, width = 11, height = 8.5)
```

Value

Called for its side effect (draws text on the current device). Returns `invisible(NULL)`.

See Also

`save_ppt()`, `theme_hv_manuscript()`

Examples

```
# --- Basic use after a base-R plot -----
plot(1:10, main = "Example")
make_footnote("examples/basic.R")

# --- With a ggplot2 figure and manuscript theme -----

library(ggplot2)
p <- ggplot(mtcars, aes(wt, mpg)) +
  geom_point() +
  labs(title = "Motor Trend Cars") +
  theme_hv_manuscript()

# Draft: print, then annotate
print(p)
make_footnote("R/exploratory.R")

# Publication: save without footnote
ggsave(file.path(tempdir(), "fig1.pdf"), p, width = 11, height = 8.5)

# --- Custom position and no timestamp -----
plot(1:10)
make_footnote("Preliminary results", timestamp = FALSE, prefix = "")

# --- Suppress the DRAFT prefix -----
plot(1:10)
make_footnote(
  text      = paste("Analyst: J. Ehrlinger |", Sys.Date()),
  timestamp = FALSE,
  prefix    = ""
)
```

nonparametric

Nonparametric survival estimates

Description

A dataset containing nonparametric competing-risk estimates used for examples in the package vignettes: the empirical companion to [parametric](#), over the same three states (event-free, death, and stroke). Originally exported from SAS via `tp.hp.dead.sas` / `tp.np.*.sas`.

Format

A data frame with 126 rows (one per event time) and 10 columns:

iv_state Follow-up time in years (0 to about 4.9); the x-axis in the vignette plots.

sginit Percent event-free (still in the initial state).

stlinit Lower confidence limit for sginit; all NA in this extract.
stuinit Upper confidence limit for sginit; all NA in this extract.
sgdead1 Cumulative percent dead; NA at times with no death.
sgstrk1 Cumulative percent with stroke; NA at times with no stroke.
stldead1 Lower confidence limit for sgdead1 (6 time points).
studead1 Upper confidence limit for sgdead1 (6 time points).
stlstrk1 Lower confidence limit for sgstrk1 (3 time points).
stustrk1 Upper confidence limit for sgstrk1 (3 time points).

Details

Column prefixes follow the SAS naming: sg* is the estimate, stl* and stu* its lower and upper confidence limits. The limits are filled only at the handful of times where an error bar is drawn and are NA elsewhere, which is how plot.sas thins error bars.

See Also

[parametric](#), [hv_nonparametric\(\)](#)

parametric

Parametric survival estimates

Description

A dataset containing parametric competing-risk survival estimates used for examples in the package vignettes. It was originally produced by a Weibull parametric survival model fitted to an aortic-valve surgery cohort and exported from SAS via tp.hp.dead.sas.

Format

A data frame with 2001 rows (fine time grid) and 41 columns:

years Follow-up time in years.
months Follow-up time in months (years * 12).
time Follow-up time in years (same as years; retained for SAS compatibility).
lag_time Lagged follow-up time (equals time for standard output).
dt Time increment between consecutive time points.
sedeath Parametric survival estimate: freedom from death (percent).
sldeath Lower 95 % confidence limit for sedeath.
sudeath Upper 95 % confidence limit for sedeath.
hedeath Instantaneous hazard rate estimate for death.
hldeath Lower 95 % confidence limit for hedeath.
hudeath Upper 95 % confidence limit for hedeath.

vedeath Variance of the death survival estimate.

nodeath Cumulative probability of death by time t (percent).

cldeath Lower 95 % confidence limit for nodeath.

cudeath Upper 95 % confidence limit for nodeath.

no1death Identical to nodeath in this extract.

tx1death Cumulative hazard integral for death (Weibull model internal).

txdeath Cumulative hazard for death (same scale as tx1death).

sestrk Parametric survival estimate: freedom from stroke (percent).

slstrk Lower 95 % confidence limit for sestrk.

sustrk Upper 95 % confidence limit for sestrk.

hestrk Instantaneous hazard rate estimate for stroke.

hlstrk Lower 95 % confidence limit for hestrk.

hustrk Upper 95 % confidence limit for hestrk.

vestrk Variance of the stroke survival estimate.

nostrk Cumulative probability of stroke by time t (percent).

clstrk Lower 95 % confidence limit for nostrk.

custrk Upper 95 % confidence limit for nostrk.

no1strk Identical to nostrk in this extract.

tx1strk Cumulative hazard integral for stroke (Weibull model internal).

txstrk Cumulative hazard for stroke (same scale as tx1strk).

cestrk Percent free of stroke, $100 - \text{nostrk}$.

noinit Percent event-free (still in the initial state).

clinit Lower 95 % confidence limit for noinit.

cuinit Upper 95 % confidence limit for noinit.

no1init Identical to noinit in this extract.

z Log-hazard estimate for the pooled Weibull model.

sez Standard error of z.

cllz Lower 95 % confidence limit for z (log-hazard scale).

cluz Upper 95 % confidence limit for z (log-hazard scale).

check Row sum of noinit, nodeath and nostrk; about 100.

Details

The three states tracked are competing risks; at every time point the percentages in the three states sum to 100 (see check):

init Event-free (still in the initial state)

death Death

strk Stroke

Each outcome suffix maps to: se* = survival estimate; sl*/su* = lower / upper 95 % CI on the survival estimate; he* = hazard rate estimate; hl*/hu* = lower / upper 95 % CI on the hazard; ve* = variance of the survival estimate; no* = cumulative incidence (percent); cl*/cu* = lower / upper 95 % CI on the cumulative incidence; no1* = identical to no* in this extract; tx*/tx1* = cumulative hazard integral (Weibull model output, used internally).

See Also

[nonparametric](#), [hv_hazard\(\)](#)

plot.hv_alluvial	<i>Plot an hv_alluvial object</i>
------------------	-----------------------------------

Description

Draws a Sankey (alluvial) diagram using [geom_alluvium](#) and [geom_stratum](#).

Usage

```
## S3 method for class 'hv_alluvial'
plot(
  x,
  stratum_fill = "gray80",
  stratum_width = 1/4,
  flow_width = 1/6,
  alpha = 0.8,
  knot_pos = 0.4,
  show_labels = TRUE,
  show_yaxis = TRUE,
  ...
)
```

Arguments

x	An hv_alluvial object.
stratum_fill	Fill color for the stratum bars. Default "gray80".
stratum_width	Width of the stratum bars as a fraction of axis spacing. Default 1/4.
flow_width	Width of the alluvium flows. Default 1/6.
alpha	Transparency of the flows, [0, 1]. Default 0.8.
knot_pos	Curvature of the flow ribbons, [0, 1]. Default 0.4.
show_labels	Logical; if TRUE, each stratum is labeled. Default TRUE.
show_yaxis	Logical; if FALSE, the y-axis title, text, ticks, and line are blanked for a clean milestone patient-flow look (the alluvium/stratum geometry is untouched). Default TRUE (counts shown). The blanking is a theme() layer, so a <i>complete</i> theme added afterward (e.g. theme_hv_manuscript()) re-asserts the axis it styles; add the theme first and re-blank, or blank the y elements yourself after it, when you want both the house theme and a clean axis.

... Ignored; present for S3 consistency.

Value

A bare `ggplot` object.

See Also

`hv_alluvial`, `theme_hv_manuscript`

Examples

```
dta <- sample_alluvial_data(n = 300, seed = 42)
axes <- c("pre_ar", "procedure", "post_ar")

# Fill flows by procedure
plot(hv_alluvial(dta, axes = axes, y_col = "freq",
  fill_col = "procedure")) +
  ggplot2::scale_fill_manual(values = hv_palette(), name = "Procedure") +
  ggplot2::scale_color_manual(values = hv_palette(), guide = "none") +
  ggplot2::labs(y = "Patients (n)") +
  theme_hv_poster()

# Two-axis (before / after)
plot(hv_alluvial(
  dta, axes = c("pre_ar", "post_ar"), y_col = "freq",
  fill_col = "pre_ar",
  axis_labels = c("Pre-operative", "Post-operative")
)) +
  ggplot2::scale_fill_brewer(palette = "RdBu", direction = -1,
    name = "AR Grade") +
  ggplot2::scale_color_brewer(palette = "RdBu", direction = -1,
    guide = "none") +
  ggplot2::labs(y = "Patients (n)") +
  theme_hv_poster()
```

plot.hv_balance

Plot an hv_balance object

Description

Builds a bare covariate balance `ggplot2` object from an `hv_balance` data object. Each covariate appears as a labeled row; points show the standardized mean difference per group. A solid line marks zero; dotted lines mark $\pm$ threshold. Add color, shape, axis scales, and a theme with `+`.

Usage

```
## S3 method for class 'hv_balance'
plot(
  x,
  point_size = 3,
  alpha = 0.8,
  hline_linetype = "dashed",
  hline_linewidth = 0.25,
  vline_linewidth = 0.2,
  threshold_linetype = "dotted",
  ...
)
```

Arguments

x	An hv_balance object.
point_size	Passed to geom_point(). Default 3.
alpha	Point transparency in [0, 1]. Default 0.8.
hline_linetype	Linetype for horizontal covariate guides. Default "dashed".
hline_linewidth	Linewidth for horizontal guides. Default 0.25.
vline_linewidth	Linewidth for the zero reference line. Default 0.2.
threshold_linetype	Linetype for the $\pm$ threshold lines. Default "dotted".
...	Ignored; present for S3 consistency.

Details

With more than one statistic, panels sit side by side (free x scales, labeled strips); with more than one pair, each pair gets a facet row. The variance ratio panel has a solid line at 1 and dotted lines at `vr_threshold`, and the KS panel has a solid line at 0 plus `ks_threshold` when given.

Value

A bare [ggplot](#) object.

See Also

[hv_balance](#), [theme_hv_manuscript](#)

Examples

```
dta <- sample_covariate_balance_data()
cb <- hv_balance(dta)

plot(cb) +
  ggplot2::scale_color_manual(
```

```

    values = c("Before match" = "red4", "After match" = "blue3"),
    name    = NULL
  ) +
  ggplot2::labs(x = "Standardized difference (%)", y = "") +
  theme_hv_poster()

```

plot.hv_consort	<i>Render a CONSORT flow diagram</i>
-----------------	--------------------------------------

Description

Draws the grid-based diagram stored in an hv_consort object on a new graphics page.

Usage

```
## S3 method for class 'hv_consort'
plot(x, ...)
```

Arguments

x	An hv_consort from hv_consort() .
...	Ignored; present for S3 consistency.

Value

x, invisibly.

See Also

[hv_consort\(\)](#), [save_ppt\(\)](#)

plot.hv_correlation_matrix	<i>Plot an hv_correlation_matrix object</i>
----------------------------	---

Description

Bare scatter-plot matrix: points only, free scales per panel, variable names in the strips. Decorate with a theme_hv_*() theme.

Usage

```
## S3 method for class 'hv_correlation_matrix'
plot(x, alpha = 0.3, point_size = 0.6, ...)
```

Arguments

x	An hv_correlation_matrix object.
alpha	Point transparency; large cohorts need it low.
point_size	Point size.
...	Ignored.

Value

A ggplot object.

plot.hv_eda	<i>Plot an hv_eda object</i>
-------------	------------------------------

Description

Draws an exploratory data analysis plot for the variable stored in the hv_eda object. Variable type (stored in x\$meta\$var_type) determines the chart:

Usage

```
## S3 method for class 'hv_eda'
plot(
  x,
  smooth_method = "loess",
  smooth_span = 0.8,
  smooth_se = FALSE,
  loess_cutoff = 10,
  groupBars = FALSE,
  alpha = 0.4,
  interactive = NULL,
  ...
)
```

Arguments

x	An hv_eda object.
smooth_method	Smoothing method for continuous plots, passed to geom_smooth . Default "loess".
smooth_span	LOESS span. Default 0.8.
smooth_se	Logical; show confidence ribbon around smooth? Default FALSE.
loess_cutoff	Integer, if the y-variable has less unique values than this, the smooth line will not be plotted. Default 10.
groupBars	Logical; Plot as grouped bars instead of stacked? Default FALSE.
alpha	Point transparency for continuous plots. Default 0.4.

interactive Give categorical bars hover labels? NULL (the default) means yes in knitr HTML output when the ggiraph package is installed, and no otherwise; TRUE or FALSE decide it. With hover labels each bar carries a tooltip naming its category, its x bin and its count and percent of the bin, shown once the plot is rendered with [hv_girafe](#). Printed or saved, the plot is the same as without them. Continuous plots are unaffected.

... Ignored; present for S3 consistency.

Details

Continuous ("Cont") Scatter plot with a LOESS smoother overlay and a rug on the x-axis for rows where the outcome is missing. The label is the plot title; the y-axis title is a short form of it, the label's trailing parenthesised units, such as "mmHg" from "Peak Gradient (mmHg)", or else the column name.

Numeric categorical ("Cat_Num") Stacked (or filled) bar chart with counts (or proportions) per x level.

Character categorical ("Cat_Char") Same stacked bar, coloring each string level separately.

Value

A bare [ggplot](#) object.

See Also

[hv_eda](#), [theme_hv_manuscript](#)

Examples

```
dta <- sample_eda_data(n = 300, seed = 42)

# --- Ordinal categorical: percentage barplot -----
plot(hv_eda(dta, x_col = "year", y_col = "nyha",
  y_label = "Preoperative NYHA Class")) +
  ggplot2::scale_fill_brewer(
    palette = "RdBu", direction = -1,
    labels = c("1" = "I", "2" = "II", "3" = "III", "4" = "IV",
      "(Missing)" = "Missing"),
    name = "NYHA"
  ) +
  ggplot2::scale_x_discrete(breaks = seq(2005, 2020, 5)) +
  ggplot2::labs(x = "Surgery Year", y = "Count") +
  theme_hv_poster()

# --- Continuous: annotated -----
plot(hv_eda(dta, x_col = "op_years", y_col = "peak_grad",
  y_label = "Peak Gradient (mmHg)")) +
  ggplot2::scale_color_manual(values = c("steelblue"), guide = "none") +
  ggplot2::scale_x_continuous(breaks = seq(0, 15, 5)) +
  ggplot2::labs(x = "Years from First Surgery Year") +
  ggplot2::annotate("text", x = 12, y = 70,
```

```

        label = "LOESS span = 0.8",
        size = 3, color = "gray40", fontface = "italic") +
theme_hv_poster()

```

plot.hv_eda_pages	<i>Plot an hv_eda_pages object</i>
-------------------	------------------------------------

Description

Lays the section's panels out as pages of ncol by nrow panels. Nothing is printed or saved; the caller prints, saves or captions each page.

Usage

```

## S3 method for class 'hv_eda_pages'
plot(x, ncol = 4L, nrow = 4L, alpha = 0.5, ...)

```

Arguments

x	An hv_eda_pages object.
ncol, nrow	Panels across and down each page. Default 4 by 4.
alpha	Point transparency for continuous panels. Default 0.5, which keeps a dense cohort readable as a distribution.
...	Passed to plot.hv_eda() , for example smooth_span, or interactive to give the bars of a percent or count page hover labels; render such a page with hv_girafe() .

Details

On a percent or count page the panels in a row share one y axis, drawn on the leftmost panel. Each continuous panel keeps its own y axis, titled with a short form of its label (see [plot.hv_eda\(\)](#)).

Colors and themes are left to the caller, as everywhere in this package. Apply them to every panel of a page at once with patchwork's &: page & theme_hv_manuscript(base_size = 8).

Value

A list of patchwork pages, possibly empty. Each page carries the names of the variables on it as attr(page, "variables"), so a report can caption it.

See Also

[hv_eda_pages\(\)](#), [plot.hv_eda\(\)](#)

Examples

```

dta <- sample_eda_data(n = 300, seed = 42)
pages <- plot(hv_eda_pages(dta, section = "count"), ncol = 3, nrow = 2)
for (page in pages) message(paste(attr(page, "variables"), collapse = ", "))

```

plot.hv_followup	<i>Plot an hv_followup object</i>
------------------	-----------------------------------

Description

Builds a bare goodness-of-follow-up ggplot2 object from an [hv_followup](#) data object. Each patient appears as a point at their operation year (x) and total follow-up time (y), shaped and colored by their state. An orange diagonal reference line shows the maximum possible follow-up for patients enrolled at each year. A vertical stem below each point is drawn only when `hv_followup()` was called with a positive `segment_drop`.

Usage

```
## S3 method for class 'hv_followup'
plot(
  x,
  type = c("followup", "event"),
  alpha = 0.8,
  diagonal_color = "orange",
  diagonal_linetype = "dashed",
  diagonal_linewidth = 0.6,
  ...
)
```

Arguments

<code>x</code>	An <code>hv_followup</code> object.
<code>type</code>	Which panel to produce: "followup" (default, death states) or "event" (requires <code>event_col</code> to have been supplied to hv_followup).
<code>alpha</code>	Point/segment transparency in $[0, 1]$. Default 0.8.
<code>diagonal_color</code>	Color of the diagonal reference line. Default "orange".
<code>diagonal_linetype</code>	Linetype for the diagonal. Default "dashed".
<code>diagonal_linewidth</code>	Linewidth for the diagonal. Default 0.6.
<code>...</code>	Ignored; present for S3 consistency.

Value

A bare [ggplot](#) object.

See Also

[hv_followup](#), [theme_hv_manuscript](#)

Examples

```
dta <- sample_goodness_followup_data()
gf <- hv_followup(dta, event_col = "ev_event",
                  event_time_col = "iv_event")

# Death panel
plot(gf) +
  ggplot2::scale_color_manual(
    values = c("Alive" = "steelblue", "Dead" = "firebrick"),
    name = NULL
  ) +
  ggplot2::labs(x = "Operation Date", y = "Follow-up (years)") +
  theme_hv_poster()

# Event panel
plot(gf, type = "event") +
  ggplot2::scale_color_manual(
    values = c("No event" = "blue", "Non-fatal event" = "green3",
              "Death" = "red"),
    name = NULL
  ) +
  ggplot2::labs(x = "Operation Date", y = "Follow-up (years)") +
  theme_hv_poster()

# Legacy stem below each point -- restored with a positive segment_drop
plot(hv_followup(dta, segment_drop = 0.2))
```

plot.hv_followup_panels

Plot an hv_followup_panels object

Description

Draws one bare goodness-of-follow-up figure per panel, through `plot.hv_followup()`. Colors, shapes, labels and themes are left to the caller, as everywhere in this package; the state levels are "Alive" and "Dead" for a death panel, and "No event", the event's label and "Death" for an event panel.

Usage

```
## S3 method for class 'hv_followup_panels'
plot(x, alpha = 0.5, ...)
```

Arguments

x	An hv_followup_panels object.
alpha	Point transparency. Default 0.5, which keeps a dense cohort readable as a distribution.
...	Passed to <code>plot.hv_followup()</code> , for example <code>diagonal_color</code> .

Value

A named list of bare ggplot objects, one per panel, in the order of x\$data.

See Also

`hv_followup_panels()`, `plot.hv_followup()`

Examples

```
dta <- sample_goodness_followup_data()
fp <- hv_followup_panels(dta, origin_year = 1990,
                        panels = list(all = list(status = "dead", time = "iv_dead")))
plots <- plot(fp)
plots$all +
  ggplot2::scale_color_manual(values = c(Alive = "#377EB8", Dead = "#E41A1C")) +
  ggplot2::labs(x = "Year of operation", y = "Follow-up (years)")
```

plot.hv_hazard

Plot an hv_hazard object

Description

Renders a bare `ggplot2::ggplot()` from an `hv_hazard()` object. Compose with `scale_color_*`, `scale_y_continuous()`, `labs()`, and `theme_hv_manuscript()` to complete the figure.

Usage

```
## S3 method for class 'hv_hazard'
plot(
  x,
  ci_alpha = 0.2,
  line_width = 1,
  point_size = 2,
  point_shape = 1L,
  errorbar_width = 0.25,
  ...
)
```

Arguments

x	An hv_hazard object from <code>hv_hazard()</code> .
ci_alpha	Transparency of the CI ribbon. Default 0.20.
line_width	Width of the parametric curve line. Default 1.0.
point_size	Size of empirical overlay points (used when <code>emp_geom = "point"</code> was set in <code>hv_hazard()</code>). Default 2.0.
point_shape	Shape code for empirical points (1 = open circle, 0 = open square). Default 1.
errorbar_width	Width of error bars on empirical points. Default 0.25.
...	Ignored; present for S3 consistency.

Value

A `ggplot2::ggplot()` object; compose with `+` to add scales, axis limits, labels, and `theme_hv_manuscript()`.

See Also

`hv_hazard()` to build the data object, `theme_hv_manuscript()` for the publication theme.

Other Hazard plot: `hv_hazard()`

`plot.hv_longitudinal` *Plot an hv_longitudinal object*

Description

Draws either a grouped bar chart of counts by time point (`type = "plot"`) or a numeric text-table panel suitable for composing below the bar chart via **patchwork** (`type = "table"`).

Usage

```
## S3 method for class 'hv_longitudinal'
plot(
  x,
  type = c("plot", "table"),
  position = "dodge",
  label_format = NULL,
  ...
)
```

Arguments

<code>x</code>	An <code>hv_longitudinal</code> object.
<code>type</code>	Which panel to produce: <code>"plot"</code> (default) or <code>"table"</code> .
<code>position</code>	Bar position for <code>type = "plot"</code> : <code>"dodge"</code> (default) or <code>"stack"</code> .
<code>label_format</code>	Formatting function applied to count values for <code>type = "table"</code> . <code>NULL</code> (default) auto-selects <code>scales::comma</code> when the scales package is installed, otherwise falls back to <code>base::as.character</code> . Pass <code>identity</code> to display counts without formatting.
<code>...</code>	Ignored; present for S3 consistency.

Value

A bare `ggplot` object.

See Also

`hv_longitudinal`

Examples

```
library(ggplot2)
dta <- sample_longitudinal_counts_data(n_patients = 300, seed = 42L)
lc <- hv_longitudinal(dta)

# Bar chart
plot(lc, type = "plot") +
  scale_fill_manual(
    values = c(Patients = "steelblue", Measurements = "firebrick"),
    name = NULL
  ) +
  labs(x = "Follow-up Window", y = "Count (n)") +
  theme_hv_poster()

# Text table panel
plot(lc, type = "table") +
  scale_color_manual(
    values = c(Patients = "steelblue", Measurements = "firebrick"),
    guide = "none"
  ) +
  theme_hv_poster()

# Compose with patchwork
# p_bar <- plot(lc, type = "plot") + <decorators>
# p_table <- plot(lc, type = "table") + <decorators>
# p_bar / p_table + patchwork::plot_layout(heights = c(3, 1))
```

plot.hv_mirror_hist	<i>Plot an hv_mirror_hist object</i>
---------------------	--------------------------------------

Description

Builds a bare mirrored-histogram ggplot2 object from an [hv_mirror_hist](#) data object. Bars for the treated group appear above the x-axis; bars for the control group appear below. Matched or weighted patients are shown in a contrasting shade. Compose with `+` to add color scales, axis labels, and [theme_hv_manuscript](#).

Usage

```
## S3 method for class 'hv_mirror_hist'
plot(x, alpha = 0.8, ...)
```

Arguments

<code>x</code>	An <code>hv_mirror_hist</code> object from hv_mirror_hist .
<code>alpha</code>	Bar transparency in $[0, 1]$. Default 0.8.
<code>...</code>	Ignored; present for S3 consistency.

Value

A bare `ggplot` object; compose with `+` to add color scales, axis limits, labels, and `theme_hv_manuscript`.

See Also

`hv_mirror_hist` to build the data object, `theme_hv_manuscript` for the publication theme, `sample_mirror_histogram_data` for example data.

Other Propensity Score & Matching: `hv_mirror_hist()`, `hv_rmst_contrast()`, `hv_rmst_curves()`, `plot.hv_rmst_contrast()`, `plot.hv_rmst_curves()`, `sample_covariate_balance_data()`

Examples

```
dta <- sample_mirror_histogram_data(n = 500)
mh  <- hv_mirror_hist(dta)

plot(mh) +
  ggplot2::labs(x = "Propensity Score (%)", y = "Count") +
  theme_hv_poster()
```

plot.hv_nnt

Plot an hv_nnt object

Description

Renders a bare `ggplot2::ggplot()` NNT (or ARR) curve from an `hv_nnt()` object.

Usage

```
## S3 method for class 'hv_nnt'
plot(x, ci_alpha = 0.2, line_width = 1, ...)
```

Arguments

<code>x</code>	An <code>hv_nnt</code> object from <code>hv_nnt()</code> .
<code>ci_alpha</code>	Transparency of the CI ribbon. Default <code>0.20</code> .
<code>line_width</code>	Line width. Default <code>1.0</code> .
<code>...</code>	Ignored.

Value

A `ggplot2::ggplot()` object.

See Also

`hv_nnt()`, `hv_survival_difference()`, `theme_hv_manuscript()`

plot.hv_nonparametric *Plot an hv_nonparametric object*

Description

Draws a smooth predicted curve with optional CI ribbon and binned data summary point overlay.

Usage

```
## S3 method for class 'hv_nonparametric'
plot(
  x,
  ci_alpha = 0.2,
  line_width = 1,
  point_size = 2.5,
  point_shape = 20L,
  ...
)
```

Arguments

x	An hv_nonparametric object.
ci_alpha	Transparency of the confidence ribbon. Default 0.2.
line_width	Width of the predicted curve line. Default 1.0.
point_size	Size of binned data summary points. Default 2.5.
point_shape	Integer shape code for data summary points. Default 20 (filled circle).
...	Ignored; present for S3 consistency.

Value

A bare [ggplot](#) object; compose with + to add scales, axis limits, labels, and [theme_hv_manuscript](#).

See Also

[hv_nonparametric](#) to build the data object, [theme_hv_manuscript](#) for the publication theme.

Other Nonparametric curves: [hv_nonparametric\(\)](#)

Examples

```
dat_two <- sample_nonparametric_curve_data(
  n = 400, time_max = 7,
  groups = c("Ozaki" = 0.7, "CE-Pericardial" = 1.3),
  outcome_type = "continuous"
)
dat_two_pts <- sample_nonparametric_curve_points(
  n = 400, time_max = 7,
```

```

    groups = c("Ozaki" = 0.7, "CE-Pericardial" = 1.3),
    outcome_type = "continuous"
  )
  np <- hv_nonparametric(dat_two, group_col = "group",
                        lower_col = "lower", upper_col = "upper",
                        data_points = dat_two_pts)

  plot(np) +
    ggplot2::scale_color_manual(
      values = c("Ozaki" = "steelblue", "CE-Pericardial" = "firebrick"),
      name = "Procedure"
    ) +
    ggplot2::scale_fill_manual(
      values = c("Ozaki" = "steelblue", "CE-Pericardial" = "firebrick"),
      guide = "none"
    ) +
    ggplot2::labs(x = "Years", y = "AV Peak Gradient (mmHg)") +
    theme_hv_poster()

```

plot.hv_ordinal	<i>Plot an hv_ordinal object</i>
-----------------	----------------------------------

Description

Draws grade-specific probability curves with an optional binned data summary point overlay.

Usage

```

## S3 method for class 'hv_ordinal'
plot(x, line_width = 1, point_size = 2.5, point_shape = 20L, ...)

```

Arguments

x	An hv_ordinal object.
line_width	Width of grade-specific curve lines. Default 1.0.
point_size	Size of binned data summary points. Default 2.5.
point_shape	Integer shape for summary points. Default 20.
...	Ignored; present for S3 consistency.

Value

A bare [ggplot](#) object.

See Also

[hv_ordinal](#), [theme_hv_manuscript](#)

Examples

```

dat <- sample_nonparametric_ordinal_data(
  n = 800, time_max = 5,
  grade_labels = c("None", "Mild", "Moderate", "Severe")
)

# Curves only, with RColorBrewer palette
plot(hv_ordinal(dat)) +
  ggplot2::scale_color_brewer(palette = "RdBu", direction = -1,
                              name = "AR Grade") +
  ggplot2::scale_x_continuous(breaks = 0:5) +
  ggplot2::scale_y_continuous(labels = scales::percent) +
  ggplot2::labs(x = "Years after Surgery", y = "Prevalence") +
  theme_hv_poster()

# Subset: show only severe grade
plot(hv_ordinal(dat[dat$grade == "Severe", ])) +
  ggplot2::scale_color_manual(values = c(Severe = "firebrick"),
                              guide = "none") +
  ggplot2::scale_y_continuous(limits = c(0, 0.25),
                              labels = scales::percent) +
  ggplot2::labs(x = "Years", y = "P(Severe TR grade)") +
  theme_hv_poster()

```

plot.hv_rmst_contrast *Plot an hv_rmst_contrast object*

Description

Builds a bare dot-and-whisker ggplot2 object from an `hv_rmst_contrast()` object. Estimators run down the y axis with the first level at the top, so the plot reads in the order supplied. A solid vertical line marks zero (no difference). Add colors, scales and a theme with `+`.

Usage

```

## S3 method for class 'hv_rmst_contrast'
plot(
  x,
  point_size = 3,
  linewidth = 0.6,
  zero_linetype = "solid",
  zero_linewidth = 0.3,
  x_label = "RMST difference (days)",
  ...
)

```

Arguments

<code>x</code>	An <code>hv_rmst_contrast</code> object.
<code>point_size</code>	Point size passed to <code>geom_pointrange()</code> . Default 3.
<code>linewidth</code>	Whisker line width. Default 0.6.
<code>zero_linetype</code>	Linetype of the zero reference line. Default "solid".
<code>zero_linewidth</code>	Line width of the zero reference line. Default 0.3.
<code>x_label</code>	X axis label. Default "RMST difference (days)".
<code>...</code>	Ignored; present for S3 consistency.

Value

A bare `ggplot2::ggplot()` object.

See Also

[hv_rmst_contrast\(\)](#)

Other Propensity Score & Matching: [hv_mirror_hist\(\)](#), [hv_rmst_contrast\(\)](#), [hv_rmst_curves\(\)](#), [plot.hv_mirror_hist\(\)](#), [plot.hv_rmst_curves\(\)](#), [sample_covariate_balance_data\(\)](#)

Examples

```
est <- data.frame(
  estimator = c("IPTW", "Overlap", "Matching"),
  diff_days = c(41, 35, 28),
  lo_days   = c(-5, 2, -14),
  hi_days   = c(87, 68, 70)
)
plot(hv_rmst_contrast(est), x_label = "RMST difference at 5 years (days)")
```

`plot.hv_rmst_curves` *Plot an `hv_rmst_curves` object*

Description

Builds a bare `ggplot2` object from an [hv_rmst_curves\(\)](#) object: weighted Kaplan-Meier step curves, one facet per estimator, colored by arm. When `tau` was supplied, the area under each curve up to `tau` is shaded (`geom_ribbon()` over a staircase polygon, so the shading follows the steps exactly) and a vertical line marks `tau`. When `estimates` was supplied, the top right of each facet carries the RMST difference and interval. Color and fill both map to arm: set them together with `scale_color_*`() and `scale_fill_*`() .

Usage

```
## S3 method for class 'hv_rmst_curves'
plot(
  x,
  shade_alpha = 0.2,
  linewidth = 0.6,
  tau_linetype = "dashed",
  label_size = 3,
  ...
)
```

Arguments

x	An hv_rmst_curves object.
shade_alpha	Transparency of the shaded area. Default 0.2.
linewidth	Step line width. Default 0.6.
tau_linetype	Linetype of the vertical tau line. Default "dashed".
label_size	Text size of the facet annotations. Default 3.
...	Ignored; present for S3 consistency.

Value

A bare `ggplot2::ggplot()` object.

See Also

[hv_rmst_curves\(\)](#)

Other Propensity Score & Matching: [hv_mirror_hist\(\)](#), [hv_rmst_contrast\(\)](#), [hv_rmst_curves\(\)](#), [plot.hv_mirror_hist\(\)](#), [plot.hv_rmst_contrast\(\)](#), [sample_covariate_balance_data\(\)](#)

Examples

```
curves <- data.frame(
  estimator = rep(c("IPTW", "Overlap"), each = 6),
  arm       = rep(rep(c("treated", "control"), each = 3), 2),
  time      = rep(c(0, 200, 500), 4),
  surv      = c(1, .9, .7, 1, .8, .5, 1, .92, .75, 1, .85, .6)
)
plot(hv_rmst_curves(curves, tau = 365), shade_alpha = 0.3)
```

plot.hv_sankey	<i>Plot an hv_sankey object</i>
----------------	---------------------------------

Description

Draws a cluster stability Sankey diagram using **ggsankey** geoms. **Requires the ggsankey package.** Install with:

```
remotes::install_github("davidsjoberg/ggsankey")
```

Usage

```
## S3 method for class 'hv_sankey'
plot(
  x,
  flow_alpha = 0.5,
  label_alpha = 0.3,
  label_size = 8,
  label_hjust = -0.05,
  group_labels = NULL,
  alpha = NULL,
  ...
)
```

Arguments

x	An hv_sankey object.
flow_alpha	Transparency of the flow bands and the dashed column guides, [0, 1]. Default 0.5.
label_alpha	Transparency of the node-label fill, [0, 1]. Default 0.3 (a light tint behind black text).
label_size	Font size for node labels in points. Default 8.
label_hjust	Horizontal justification offset for node labels. Default -0.05.
group_labels	Optional named character vector mapping a cluster_cols value to a milestone label. When supplied, that column's x-axis tick shows "<col>\n<label>"; unlisted columns show the bare column name. Default NULL (bare column names).
alpha	Deprecated. If supplied, sets both flow_alpha and label_alpha (back-compatibility) and emits a message steering you to the two new arguments. Default NULL.
...	Ignored; present for S3 consistency.

Value

A **ggplot** object using **ggsankey** geoms. Compose with `scale_fill_manual()`, `labs()`, `theme()`, and `theme_hv_manuscript`.

See Also

[hv_sankey](#), [theme_hv_manuscript](#)

Examples

```
dta <- sample_cluster_sankey_data(n = 300, seed = 42)

if (requireNamespace("ggsankey", quietly = TRUE)) {
  plot(hv_sankey(dta)) +
    ggplot2::labs(x = NULL, title = "Cluster Stability: K = 2 to 9") +
    theme_hv_poster()

  # Subset to K = 2 to 6
  plot(hv_sankey(dta, cluster_cols = paste0("C", 2:6))) +
    ggplot2::labs(x = NULL) +
    theme_hv_poster()
}
```

plot.hv_spaghetti	<i>Plot an hv_spaghetti object</i>
-------------------	------------------------------------

Description

Draws one trajectory line per subject over time, optionally stratified by color and with a LOESS (or other) smooth overlay.

Usage

```
## S3 method for class 'hv_spaghetti'
plot(
  x,
  line_colour = "gray50",
  line_width = 0.2,
  alpha = 0.8,
  add_smooth = FALSE,
  smooth_method = "loess",
  smooth_se = FALSE,
  smooth_width = 1.2,
  y_labels = NULL,
  ...,
  line_color = line_colour
)
```

Arguments

<code>x</code>	An <code>hv_spaghetti</code> object.
<code>line_colour</code>	The same as <code>line_color</code> : an alias kept for existing code. Give one or the other; both with different values is an error.
<code>line_width</code>	Line width for individual trajectories. Default 0.2.
<code>alpha</code>	Transparency of plot elements in [0, 1]. Default 0.8.
<code>add_smooth</code>	Logical; overlay a smoother? Default FALSE.
<code>smooth_method</code>	Smoothing method passed to <code>geom_smooth()</code> . Default "loess".
<code>smooth_se</code>	Logical; show CI ribbon around smooth? Default FALSE.
<code>smooth_width</code>	Line width for the smooth overlay. Default 1.2.
<code>y_labels</code>	Named numeric vector for an ordinal y-axis, e.g. <code>c(None = 0, Mild = 1, Moderate = 2, Severe = 3)</code> . Default NULL (standard numeric axis).
<code>...</code>	Ignored; present for S3 consistency.
<code>line_color</code>	Fixed line color used when the object has no <code>color_col</code> . Default "gray50".

Value

A bare `ggplot` object; compose with `+` to add scales, axis limits, labels, and `theme_hv_manuscript`.

See Also

`hv_spaghetti` to build the data object, `theme_hv_manuscript` for the publication theme.

Other Spaghetti plot: `hv_spaghetti()`

Examples

```
dta <- sample_spaghetti_data(n_patients = 150, seed = 42)
sp <- hv_spaghetti(dta, color_col = "group")

# With LOESS smooth overlay
plot(sp, add_smooth = TRUE) +
  ggplot2::scale_color_manual(values = hv_palette(), name = NULL) +
  ggplot2::labs(x = "Years", y = "AV Mean Gradient (mmHg)") +
  theme_hv_poster()

# Ordinal y-axis
dta_ord <- dta
dta_ord$value <- round(pmin(3, pmax(0, dta$value / 12)))
plot(hv_spaghetti(dta_ord, color_col = "group"),
     y_labels = c(None = 0, Mild = 1, Moderate = 2, Severe = 3)) +
  ggplot2::labs(x = "Years", y = "MR Grade") +
  theme_hv_poster()

# --- Global theme (set once per session) -----
old <- ggplot2::theme_set(theme_hv_manuscript())
plot(sp, add_smooth = TRUE) +
```

```
ggplot2::scale_color_manual(values = hv_palette(), name = NULL) +
  ggplot2::labs(x = "Years", y = "AV Mean Gradient (mmHg)")
ggplot2::theme_set(old)
```

```
# See vignette("plot-decorators", package = "hvtiPlotR") for theming,
# color scales, annotation labels, and saving plots.
```

plot.hv_stacked	<i>Plot an hv_stacked object</i>
-----------------	----------------------------------

Description

Draws a stacked (or proportional fill) histogram for the grouped numeric variable stored in the hv_stacked object.

Usage

```
## S3 method for class 'hv_stacked'
plot(x, ...)
```

Arguments

x	An hv_stacked object.
...	Ignored; present for S3 consistency.

Value

A bare [ggplot](#) object. Add scales, labels, and themes with the usual + operator.

See Also

[hv_stacked](#), [theme_hv_manuscript](#)

Examples

```
dta <- sample_stacked_histogram_data()

# Count histogram
plot(hv_stacked(dta, x_col = "year", group_col = "category")) +
  ggplot2::scale_fill_manual(values = hv_palette(), name = "Category") +
  ggplot2::scale_color_manual(values = hv_palette(), name = "Category") +
  ggplot2::labs(x = "Year", y = "Count") +
  theme_hv_poster()

# Proportional (fill) histogram with manual colors
plot(hv_stacked(dta, x_col = "year", group_col = "category",
  position = "fill")) +
```

```

ggplot2::scale_fill_manual(
  values = c("1" = "pink", "2" = "cyan", "3" = "orangered"),
  labels = c("1" = "Group A", "2" = "Group B", "3" = "Group C"),
  name = "Category"
) +
ggplot2::scale_color_manual(
  values = c("1" = "pink", "2" = "cyan", "3" = "orangered"),
  guide = "none"
) +
ggplot2::labs(x = "Year", y = "Proportion") +
theme_hv_poster()

```

plot.hv_survival	<i>Plot an hv_survival object</i>
------------------	-----------------------------------

Description

Builds a bare ggplot2 object from an [hv_survival](#) data object. The plot contains the correct aesthetics and geometries but no scale, label, or theme modifications; add those with + as you would with any ggplot2 object.

Usage

```

## S3 method for class 'hv_survival'
plot(
  x,
  type = c("survival", "cumhaz", "hazard", "loglog", "life"),
  conf_int = TRUE,
  alpha = 0.8,
  ...
)

```

Arguments

x	An hv_survival object.
type	Which plot variant to produce. One of: "survival" (default, PLOTS=1) KM step function with optional CI ribbon; y-axis on the 0–100 percent scale. "cumhaz" (PLOT=1) Nelson-Aalen cumulative hazard $H(t) = -\log S(t)$. "hazard" (PLOTH=1) Instantaneous hazard $h(t)$; add geom_smooth(method="loess") for a smoothed publication curve. "loglog" Log-log diagnostic: $\log H(t)$ vs $\log t$. Parallel lines across strata support proportional hazards. "life" (PLOT=1) Restricted mean survival time (integral of $S(t)$) vs time.
conf_int	Logical; draw a CI ribbon on the "survival" plot. Default TRUE. Ignored for other type values.

alpha Line/point transparency in [0, 1]. Default 0.8.
 ... Ignored; present for S3 consistency.

Value

A bare `ggplot` object; compose with `+` to add scales, axis limits, labels, and `theme_hv_manuscript`.

See Also

`hv_survival` to build the data object, `theme_hv_manuscript` for the publication theme.

Other Kaplan-Meier survival: `hv_survival()`, `sample_survival_data()`

Examples

```
dta <- sample_survival_data(n = 500, seed = 42)
km <- hv_survival(dta)

# Default survival curve
plot(km) +
  ggplot2::labs(x = "Years after Operation", y = "Survival (%)") +
  theme_hv_poster()

# Cumulative hazard
plot(km, type = "cumhaz") +
  ggplot2::labs(x = "Years", y = "Cumulative Hazard") +
  theme_hv_poster()

# Hazard rate with loess smoother
plot(km, type = "hazard") +
  ggplot2::geom_smooth(
    ggplot2::aes(x = .data[["mid_time"]], y = .data[["hazard"]]),
    method = "loess", se = FALSE, span = 0.5
  ) +
  ggplot2::labs(x = "Years", y = "Instantaneous Hazard") +
  theme_hv_poster()
```

plot.hv_survival_difference

Plot an hv_survival_difference object

Description

Renders a bare `ggplot2::ggplot()` survival-difference curve. Compose with `geom_hline(yintercept = 0)`, `scale_y_continuous()`, `labs()`, and `theme_hv_manuscript()`.

Usage

```
## S3 method for class 'hv_survival_difference'
plot(x, ci_alpha = 0.2, line_width = 1, ...)
```

Arguments

x	An hv_survival_difference object.
ci_alpha	Transparency of the CI ribbon. Default 0.20.
line_width	Line width. Default 1.0.
...	Ignored.

Value

A `ggplot2::ggplot()` object.

See Also

[hv_survival_difference\(\)](#), [hv_nnt\(\)](#), [theme_hv_manuscript\(\)](#)

plot.hv_trends	<i>Plot an hv_trends object</i>
----------------	---------------------------------

Description

Draws a LOESS smooth overlaid with per-x-value summary-statistic points (mean or median computed at construction time).

Usage

```
## S3 method for class 'hv_trends'
plot(
  x,
  smoother = "loess",
  span = 0.75,
  se = FALSE,
  point_size = 2.5,
  point_shape = 19L,
  alpha = 0.2,
  ...
)
```

Arguments

x	An hv_trends object.
smoother	Smoothing method passed to geom_smooth . Default "loess".
span	Span for LOESS smoother. Default 0.75.
se	Logical; show confidence ribbon around smooth? Default FALSE.
point_size	Size of the annual summary points. Default 2.5.
point_shape	Integer shape code for the summary points (single-group only; ignored when group_col is set; use <code>scale_shape_manual()</code> instead). Default 19L.
alpha	Transparency of the smooth ribbon when se = TRUE. Default 0.2.
...	Ignored; present for S3 consistency.

Value

A bare `ggplot` object; compose with `+` to add scales, axis limits, labels, and `theme_hv_manuscript`.

See Also

`hv_trends` to build the data object, `theme_hv_manuscript` for the publication theme.

Other Temporal trends: `hv_trends()`

Examples

```
# --- tp.rp.trends.sas: single continuous outcome, 1968-2000 -----
one <- sample_trends_data(n = 600, year_range = c(1968L, 2000L),
  groups = NULL)
plot(hv_trends(one, group_col = NULL)) +
  ggplot2::scale_x_continuous(limits = c(1968, 2000),
    breaks = seq(1968, 2000, 4)) +
  ggplot2::scale_y_continuous(limits = c(0, 10),
    breaks = seq(0, 10, 2)) +
  ggplot2::labs(x = "Year", y = "Cases/year") +
  theme_hv_poster()

# --- tp.lp.trends.sas: binary % outcomes, 1970-2000 by 10 -----
dta_lp <- sample_trends_data(
  n = 800, year_range = c(1970L, 2000L),
  groups = c("Shock %", "Pre-op IABP %", "Inotropes %"))
plot(hv_trends(dta_lp)) +
  ggplot2::scale_color_manual(
    values = c("Shock %" = "steelblue", "Pre-op IABP %" = "firebrick",
      "Inotropes %" = "forestgreen"), name = NULL) +
  ggplot2::scale_x_continuous(limits = c(1970, 2000),
    breaks = seq(1970, 2000, 10)) +
  ggplot2::scale_y_continuous(limits = c(0, 100),
    breaks = seq(0, 100, 10)) +
  ggplot2::coord_cartesian(xlim = c(1970, 2000), ylim = c(0, 100)) +
  ggplot2::labs(x = "Year", y = "Percent (%)") +
  theme_hv_poster()

# --- tp.lp.trends.age.sas: age on x-axis, 25-85 by 10 -----
dta_age <- sample_trends_data(
  n = 600, year_range = c(25L, 85L),
  groups = c("Repair %", "Bioprosthesis %"), seed = 7L)
plot(hv_trends(dta_age)) +
  ggplot2::scale_x_continuous(limits = c(25, 85),
    breaks = seq(25, 85, 10)) +
  ggplot2::scale_y_continuous(limits = c(0, 100),
    breaks = seq(0, 100, 20)) +
  ggplot2::coord_cartesian(xlim = c(25, 85), ylim = c(0, 100)) +
  ggplot2::labs(x = "Age (years)", y = "Percent (%)") +
  theme_hv_poster()

# --- tp.lp.trends.polytomous.sas: repair types, 1990-1999 by 1 -----
dta_poly <- sample_trends_data(
```

```

n = 800, year_range = c(1990L, 1999L),
groups = c("CE", "Cosgrove", "Periguard", "DeVega"), seed = 5L)
plot(hv_trends(dta_poly)) +
  ggplot2::scale_color_manual(
    values = c(CE = "steelblue", Cosgrove = "firebrick",
              Periguard = "forestgreen", DeVega = "goldenrod3"),
    name = "Repair type") +
  ggplot2::scale_x_continuous(limits = c(1990, 1999),
                             breaks = seq(1990, 1999, 1)) +
  ggplot2::scale_y_continuous(limits = c(0, 100),
                             breaks = seq(0, 100, 10)) +
  ggplot2::coord_cartesian(xlim = c(1990, 1999), ylim = c(0, 100)) +
  ggplot2::labs(x = "Year", y = "Percent (%)") +
  theme_hv_poster()

# --- Save -----

tr <- hv_trends(
  sample_trends_data(n = 800, year_range = c(1985L, 2015L),
                  groups = c("I", "II", "III", "IV")),
  summary_fn = "median"
)
p <- plot(tr) +
  ggplot2::scale_color_manual(values = hv_palette(), name = "NYHA Class") +
  ggplot2::scale_x_continuous(limits = c(1985, 2015),
                             breaks = seq(1985, 2015, 5)) +
  ggplot2::labs(x = "Years", y = "%") +
  theme_hv_poster()
ggplot2::ggsave(file.path(tempdir(), "trends.pdf"), p,
                width = 11.5, height = 8)

# --- Global theme (set once per session) -----

old <- ggplot2::theme_set(theme_hv_manuscript())
plot(hv_trends(dta_poly)) +
  ggplot2::scale_color_manual(values = hv_palette(), name = "Repair type")
ggplot2::theme_set(old)

# See vignette("plot-decorators", package = "hvtiPlotR") for theming,
# color scales, annotation labels, and saving plots.

```

plot.hv_upset

Plot an hv_upset object

Description

Draws an UpSet plot using [scale_x_upset](#).

Usage

```
## S3 method for class 'hv_upset'
plot(
  x,
  n_intersections = 10L,
  sort_by = c("freq", "degree"),
  fill_col = NULL,
  bar_fill = "gray40",
  set_size = TRUE,
  set_size_position = c("right", "left"),
  set_size_sort = c("descending", "ascending", "none"),
  set_size_fill = "steelblue",
  width_ratio = 0.3,
  ...
)
```

Arguments

<code>x</code>	An <code>hv_upset</code> object.
<code>n_intersections</code>	Number of intersections to display, ordered by <code>sort_by</code> . Default 10. Patients whose combination falls outside the top <code>n_intersections</code> are not drawn , and ggplot2 reports them as removed rows. That is expected, not an error: raise <code>n_intersections</code> to show more of the cohort.
<code>sort_by</code>	How to order intersections: "freq" (default, by frequency) or "degree" (by number of sets in each combination).
<code>fill_col</code>	Optional column name in <code>x\$data</code> to fill the intersection bars by (stacks bars by group). Default NULL (single color, supplied via <code>bar_fill</code>).
<code>bar_fill</code>	Single fill color for the intersection bars when <code>fill_col</code> is NULL. Default "gray40".
<code>set_size</code>	Logical; if TRUE (default), compose a set-size sidebar as a patchwork. If FALSE, return only the intersection-bar ggplot.
<code>set_size_position</code>	"right" (default) or "left".
<code>set_size_sort</code>	Sort order for the sidebar: "descending" (default), "ascending", or "none" (preserve intersect order).
<code>set_size_fill</code>	Fill color for the sidebar bars. Default "steelblue".
<code>width_ratio</code>	Fraction of horizontal space given to the set-size sidebar (only used when <code>set_size</code> = TRUE). Default 0.3.
<code>...</code>	Ignored; present for S3 consistency.

Details

When `set_size` = TRUE (the default) the function composes a patchwork of two plots: a horizontal set-size sidebar and the intersection bar chart. Apply themes to all panels with patchwork's & operator:

```
plot(up) & theme_hv_poster()
```

Pass `set_size = FALSE` to get a single intersection-bar ggplot for full customization; themes then apply via `+`:

```
plot(up, set_size = FALSE) + theme_hv_poster()
```

Value

A ggplot when `set_size = FALSE` (themes apply with `+`); a patchwork composite when `set_size = TRUE` (default; themes apply with `&` to cover all sub-panels).

See Also

[hv_upset](#), [theme_hv_manuscript](#)

Examples

```
sets <- c("AV_Replacement", "AV_Repair", "MV_Replacement", "MV_Repair",
         "TV_Repair", "Aorta", "CABG")
dta <- sample_upset_data(n = 300, seed = 42)
up <- hv_upset(dta, intersect = sets)

# Default: intersection bars + set-size sidebar (patchwork composite).
# Use `&` to theme every sub-panel.
plot(up) & theme_hv_poster()

# Intersection bars only - single ggplot, themes apply with `+`.
plot(up, set_size = FALSE) + theme_hv_poster()

# Fill bars by an external grouping variable (e.g. era).
dta$era <- ifelse(seq_len(nrow(dta)) <= 150, "Early", "Recent")
up_era <- hv_upset(dta, intersect = sets)
plot(up_era, fill_col = "era", set_size = FALSE) +
  ggplot2::scale_fill_manual(
    values = c(Early = "gray60", Recent = "steelblue"),
    name = "Era"
  ) +
  theme_hv_poster()
```

plot.hv_venn

Plot an hv_venn object

Description

Draws a 2-3 set Venn diagram using `ggvenn::ggvenn()` and returns a bare **ggplot2** object you can finish with `+`.

Usage

```
## S3 method for class 'hv_venn'
plot(
  x,
  show_percentage = TRUE,
  show_counts = TRUE,
  fill = NULL,
  text_size = 4,
  set_name_size = 6,
  ...
)
```

Arguments

<code>x</code>	An <code>hv_venn</code> object.
<code>show_percentage</code>	Logical; show each region's percentage. Default <code>TRUE</code> .
<code>show_counts</code>	Logical; show each region's count. Default <code>TRUE</code> .
<code>fill</code>	Optional vector of fill colors, one per set. <code>NULL</code> (default) uses ggvenn 's palette.
<code>text_size</code>	Region label text size. Default 4.
<code>set_name_size</code>	Set name text size. Default 6.
<code>...</code>	Forwarded to <code>ggvenn::ggvenn()</code> for finer styling (unlike most <code>plot.hv_*</code> methods, which ignore <code>...</code> ; ggvenn bakes labels into its geoms, so forwarding is the only way to reach them).

Value

A `ggplot` object, already styled by **ggvenn** and coordinate-free (no axes). Tune it through this method's arguments (`fill`, `text_size`, `set_name_size`, `...`). Do *not* add an axis-bearing house theme such as `theme_hv_manuscript()`: a Venn has no meaningful x/y, and the theme would paste spurious axes onto it.

See Also

[hv_venn](#)

Examples

```
dta <- sample_upset_data(n = 300, seed = 42)
v <- hv_venn(dta, sets = c("AV_Replacement", "MV_Replacement", "CABG"))
plot(v)
```

print.hv_alluvial	<i>Print an hv_alluvial object</i>
-------------------	------------------------------------

Description

Print an hv_alluvial object

Usage

```
## S3 method for class 'hv_alluvial'  
print(x, ...)
```

Arguments

x	An hv_alluvial object from hv_alluvial .
...	Ignored.

Value

x, invisibly.

print.hv_balance	<i>Print an hv_balance object</i>
------------------	-----------------------------------

Description

Print an hv_balance object

Usage

```
## S3 method for class 'hv_balance'  
print(x, ...)
```

Arguments

x	An hv_balance object from hv_balance .
...	Ignored.

Value

x, invisibly.

<code>print.hv_consort</code>	<i>Print an hv_consort object</i>
-------------------------------	-----------------------------------

Description

Print an hv_consort object

Usage

```
## S3 method for class 'hv_consort'  
print(x, ...)
```

Arguments

<code>x</code>	An hv_consort from hv_consort() .
<code>...</code>	Ignored.

Value

`x`, invisibly.

<code>print.hv_consort_tracker</code>	<i>Print an hv_consort_tracker object</i>
---------------------------------------	---

Description

Print an hv_consort_tracker object

Usage

```
## S3 method for class 'hv_consort_tracker'  
print(x, ...)
```

Arguments

<code>x</code>	An hv_consort_tracker from hv_consort_start() .
<code>...</code>	Ignored.

Value

`x`, invisibly.

print.hv_eda	<i>Print an hv_eda object</i>
--------------	-------------------------------

Description

Print an hv_eda object

Usage

```
## S3 method for class 'hv_eda'  
print(x, ...)
```

Arguments

- x An hv_eda object from [hv_eda](#).
- ... Ignored.

Value

x, invisibly.

print.hv_eda_pages	<i>Print an hv_eda_pages object</i>
--------------------	-------------------------------------

Description

Print an hv_eda_pages object

Usage

```
## S3 method for class 'hv_eda_pages'  
print(x, ...)
```

Arguments

- x An hv_eda_pages object from [hv_eda_pages\(\)](#).
- ... Ignored.

Value

x, invisibly.

print.hv_followup	<i>Print an hv_followup object</i>
-------------------	------------------------------------

Description

Print an hv_followup object

Usage

```
## S3 method for class 'hv_followup'  
print(x, ...)
```

Arguments

x	An hv_followup object from hv_followup .
...	Ignored.

Value

x, invisibly.

print.hv_followup_panels	<i>Print an hv_followup_panels object</i>
--------------------------	---

Description

Print an hv_followup_panels object

Usage

```
## S3 method for class 'hv_followup_panels'  
print(x, ...)
```

Arguments

x	An hv_followup_panels object from hv_followup_panels() .
...	Ignored.

Value

x, invisibly.

print.hv_hazard	<i>Print an hv_hazard object</i>
-----------------	----------------------------------

Description

Print an hv_hazard object

Usage

```
## S3 method for class 'hv_hazard'  
print(x, ...)
```

Arguments

x	An hv_hazard object from hv_hazard() .
...	Ignored.

Value

x, invisibly.

print.hv_longitudinal	<i>Print an hv_longitudinal object</i>
-----------------------	--

Description

Print an hv_longitudinal object

Usage

```
## S3 method for class 'hv_longitudinal'  
print(x, ...)
```

Arguments

x	An hv_longitudinal object from hv_longitudinal .
...	Ignored.

Value

x, invisibly.

<code>print.hv_mirror_hist</code>	<i>Print an hv_mirror_hist object</i>
-----------------------------------	---------------------------------------

Description

Print an hv_mirror_hist object

Usage

```
## S3 method for class 'hv_mirror_hist'  
print(x, ...)
```

Arguments

- x An hv_mirror_hist object from [hv_mirror_hist](#).
- ... Ignored.

Value

x, invisibly.

<code>print.hv_nnt</code>	<i>Print an hv_nnt object</i>
---------------------------	-------------------------------

Description

Print an hv_nnt object

Usage

```
## S3 method for class 'hv_nnt'  
print(x, ...)
```

Arguments

- x An hv_nnt object from [hv_nnt\(\)](#).
- ... Ignored.

Value

x, invisibly.

```
print.hv_nonparametric
```

Print an hv_nonparametric object

Description

Print an hv_nonparametric object

Usage

```
## S3 method for class 'hv_nonparametric'  
print(x, ...)
```

Arguments

x	An hv_nonparametric object.
...	Ignored.

Value

x, invisibly.

```
print.hv_ordinal
```

Print an hv_ordinal object

Description

Print an hv_ordinal object

Usage

```
## S3 method for class 'hv_ordinal'  
print(x, ...)
```

Arguments

x	An hv_ordinal object from hv_ordinal .
...	Ignored.

Value

x, invisibly.

```
print.hv_rmst_contrast
```

Print an hv_rmst_contrast object

Description

Print an hv_rmst_contrast object

Usage

```
## S3 method for class 'hv_rmst_contrast'  
print(x, ...)
```

Arguments

x	An hv_rmst_contrast object from hv_rmst_contrast() .
...	Ignored.

Value

x, invisibly.

```
print.hv_rmst_curves
```

Print an hv_rmst_curves object

Description

Print an hv_rmst_curves object

Usage

```
## S3 method for class 'hv_rmst_curves'  
print(x, ...)
```

Arguments

x	An hv_rmst_curves object from hv_rmst_curves() .
...	Ignored.

Value

x, invisibly.

print.hv_sankey	<i>Print an hv_sankey object</i>
-----------------	----------------------------------

Description

Print an hv_sankey object

Usage

```
## S3 method for class 'hv_sankey'  
print(x, ...)
```

Arguments

x	An hv_sankey object from hv_sankey .
...	Ignored.

Value

x, invisibly.

print.hv_spaghetti	<i>Print an hv_spaghetti object</i>
--------------------	-------------------------------------

Description

Print an hv_spaghetti object

Usage

```
## S3 method for class 'hv_spaghetti'  
print(x, ...)
```

Arguments

x	An hv_spaghetti object from hv_spaghetti .
...	Ignored.

Value

x, invisibly.

<code>print.hv_stacked</code>	<i>Print an hv_stacked object</i>
-------------------------------	-----------------------------------

Description

Print an hv_stacked object

Usage

```
## S3 method for class 'hv_stacked'  
print(x, ...)
```

Arguments

- `x` An hv_stacked object from [hv_stacked](#).
- `...` Ignored.

Value

`x`, invisibly.

<code>print.hv_survival</code>	<i>Print an hv_survival object</i>
--------------------------------	------------------------------------

Description

Print an hv_survival object

Usage

```
## S3 method for class 'hv_survival'  
print(x, ...)
```

Arguments

- `x` An hv_survival object from [hv_survival](#).
- `...` Ignored.

Value

`x`, invisibly.

```
print.hv_survival_difference
```

Print an hv_survival_difference object

Description

Print an hv_survival_difference object

Usage

```
## S3 method for class 'hv_survival_difference'  
print(x, ...)
```

Arguments

x	An hv_survival_difference object.
...	Ignored.

Value

x, invisibly.

```
print.hv_trends
```

Print an hv_trends object

Description

Print an hv_trends object

Usage

```
## S3 method for class 'hv_trends'  
print(x, ...)
```

Arguments

x	An hv_trends object from hv_trends .
...	Ignored.

Value

x, invisibly.

<code>print.hv_upset</code>	<i>Print an hv_upset object</i>
-----------------------------	---------------------------------

Description

Print an hv_upset object

Usage

```
## S3 method for class 'hv_upset'  
print(x, ...)
```

Arguments

<code>x</code>	An hv_upset object from hv_upset .
<code>...</code>	Ignored.

Value

`x`, invisibly.

<code>print.hv_venn</code>	<i>Print an hv_venn object</i>
----------------------------	--------------------------------

Description

Print an hv_venn object

Usage

```
## S3 method for class 'hv_venn'  
print(x, ...)
```

Arguments

<code>x</code>	An hv_venn object from hv_venn .
<code>...</code>	Ignored.

Value

`x`, invisibly.

sample_alluvial_data *Sample Sankey / Alluvial Data*

Description

Generates a realistic cardiac-surgery data set suitable for demonstrating `hv_alluvial()`. Each row represents a unique combination of pre-operative AV regurgitation grade, surgical procedure type, and post-operative AV regurgitation grade, together with the patient count (`freq`) for that combination. The co-occurrence structure reflects realistic clinical patterns: more severe pre-operative disease is more likely to improve post-operatively following valve surgery.

Usage

```
sample_alluvial_data(n = 300, seed = 42L)
```

Arguments

<code>n</code>	Total number of simulated patients before aggregation. Default 300.
<code>seed</code>	Random seed for reproducibility. Default 42.

Value

A data frame with columns: `pre_ar` (factor), `procedure` (factor), `post_ar` (factor), `freq` (integer count). Rows with `freq == 0` are excluded.

See Also

[hv_alluvial\(\)](#)

Examples

```
dta <- sample_alluvial_data(n = 300, seed = 42)
head(dta)
# Axes in order: pre-op grade -> procedure -> post-op grade
with(dta, tapply(freq, list(pre_ar, post_ar), sum, default = 0))
```

sample_cluster_sankey_data *Sample Cluster Stability Sankey Data*

Description

Generates a synthetic dataset with one row per patient and columns C2–C9 holding letter-labeled cluster assignments at successive values of K (number of clusters). The hierarchical merge structure follows the pattern from the HVTI PAM clustering analysis:

Usage

```
sample_cluster_sankey_data(  
  n = 300L,  
  probs = c(B = 0.18, F = 0.12, H = 0.06, D = 0.12, I = 0.04, C = 0.14, E = 0.11, G =  
    0.08, A = 0.15),  
  seed = 42L  
)
```

Arguments

- n Number of patients. Default 300.
- probs Named numeric vector of C9-level cluster probabilities (must sum to 1), in the order c(B, F, H, D, I, C, E, G, A). Default uses approximate equal-area proportions.
- seed Random seed. Default 42L.

Details

C9 label	C2	C3	C4	C5	C6	C7	C8	C9
A	A	A	A	A	A	A	A	A
B	B	B	B	B	B	B	B	B
C	A	C	C	C	C	C	C	C
D	B	B	D	D	D	D	D	D
E	A	C	C	E	E	E	E	E
F	B	B	B	B	F	F	F	F
G	A	A	A	A	A	G	G	G
H	B	B	B	B	F	F	H	H
I	B	B	D	D	D	D	D	I

Value

A data frame with n rows and columns C2–C9, each a factor ordered by the hierarchical cluster labels.

See Also

```
hv_sankey()
```

Examples

```
dta <- sample_cluster_sankey_data(n = 200, seed = 42)  
head(dta)  
table(dta$C9)
```

sample_consort_data	<i>Generate a sample CONSORT tracker for demos and testing</i>
---------------------	--

Description

Simulates a cardiac surgery cohort and builds a three-stage hv_consort_tracker suitable for testing [hv_consort\(\)](#) and demonstrating the tracker API.

Usage

```
sample_consort_data(n = 300L, seed = 42L)
```

Arguments

n	Integer. Total number of simulated patients. Default 300.
seed	Integer random seed for reproducibility. Default 42.

Value

An hv_consort_tracker with three stages: *Screened* -> *Eligible* (excl: age < 18, no STS procedure) -> *Analyzed* (excl: missing echocardiogram, prior trial).

See Also

[hv_consort_start\(\)](#), [hv_consort\(\)](#)

Examples

```
tracker <- sample_consort_data()
print(tracker)
hv_consort_summary(tracker)

hv_consort(tracker) |> plot()
```

sample_correlation_data	<i>Sample data for hv_correlation_matrix()</i>
-------------------------	--

Description

Simulated preoperative labs with a built-in dependence on HbA1c, shaped like the correlation job it stands in for.

Usage

```
sample_correlation_data(n = 300, seed = 42)
```

Arguments

n	Number of patients.
seed	Random seed.

Value

A data frame with numeric a1c, glucose, creatinine, albumin, and a three-level factor a1c_grp.

Examples

```
head(sample_correlation_data(n = 10))
```

```
sample_covariate_balance_data
```

Generate Sample Covariate Balance Data

Description

Produces a reproducible long-format data frame suitable for testing and demonstrating [hv_balance\(\)](#). Rather than drawing SMDs from independent normals, this generator simulates patient-level covariates through a logistic propensity score model, computes group standardized mean differences before matching, then performs greedy 1:1 nearest-neighbor caliper matching and computes residual differences in the matched cohort.

Usage

```
sample_covariate_balance_data(
  n_vars = 12,
  n = 600,
  separation = 1.5,
  caliper = 0.05,
  group_levels = c("Before match", "After match"),
  seed = 42L
)
```

Arguments

n_vars	Integer; number of covariates to generate. Default 12.
n	Integer; total number of simulated patients before matching. Default 600 (roughly 300 per group at separation = 1.5).

separation	Numeric; distance between the two group means on the log-odds scale. Larger values push propensity score distributions further apart, increasing the proportion of unmatched extreme-score patients and residual imbalance after matching. Default 1.5.
caliper	Matching caliper expressed in propensity-score units (0–1 scale). Patients without a partner within this distance are left unmatched. Default 0.05.
group_levels	Length-2 character vector of group labels. Default c("Before match", "After match").
seed	Integer random seed for reproducibility. Default 42.

Details

The result captures the pattern seen in real studies: covariates that drive treatment selection show large imbalance before matching; matching substantially reduces imbalance, but patients at the propensity score extremes cannot be matched, leaving small residual differences for the strongest confounders.

Value

A data frame with $2 * n_vars$ rows and columns variable, group, and std_diff (standardized mean difference as a percentage).

See Also

Other Propensity Score & Matching: [hv_mirror_hist\(\)](#), [hv_rmst_contrast\(\)](#), [hv_rmst_curves\(\)](#), [plot.hv_mirror_hist\(\)](#), [plot.hv_rmst_contrast\(\)](#), [plot.hv_rmst_curves\(\)](#)

Examples

```
dta <- sample_covariate_balance_data()
head(dta)

# Higher separation -> more unmatched extremes -> more residual imbalance
dta2 <- sample_covariate_balance_data(
  n_vars      = 8,
  separation   = 2.0,
  group_levels = c("Unweighted", "IPTW weighted")
)
```

sample_eda_data

Sample EDA Data

Description

Generates a realistic mixed-type patient-level data frame for demonstrating [hv_eda\(\)](#) and [eda_select_vars\(\)](#). The data mimics a cardiac surgery registry with binary, ordinal, character-categorical, and continuous variables, plus a modest proportion of missing values.

Usage

```
sample_eda_data(n = 300L, year_range = c(2005L, 2020L), seed = 42L)
```

Arguments

n	Number of patients. Default 300.
year_range	Integer vector c(start, end) for surgery years. Default c(2005L, 2020L).
seed	Random seed for reproducibility. Default 42.

Value

A data frame with columns:

- year: integer surgery year (discrete x for barplots)
- op_years: continuous years from first year in range (x for scatterplots)
- male: binary 0/1 (sex)
- cabg: binary 0/1 (concomitant CABG)
- nyha: ordinal 1–4 (NYHA class)
- valve_morph: character (valve morphology: Bicuspid / Tricuspid / Unicuspid)
- ef: continuous ejection fraction (%)
- lv_mass: continuous LV mass index (g/m²)
- peak_grad: continuous peak gradient (mmHg)

See Also

[hv_eda\(\)](#), [eda_classify_var\(\)](#), [eda_select_vars\(\)](#)

Examples

```
dta <- sample_eda_data()
head(dta)
sapply(dta, eda_classify_var)
```

sample_goodness_followup_data

Generate Sample Goodness-of-Follow-Up Data

Description

Produces a reproducible data frame suitable for testing and demonstrating [hv_followup\(\)](#). Operation dates are drawn uniformly over the study period; death and non-fatal event times are simulated from exponential distributions and censored at each patient's potential follow-up. The deads column approximates active/systematic death ascertainment by restricting to deaths within 90% of the potential follow-up window, mirroring the distinction between dead and deads in the legacy tp.dp.gfup.R template.

Usage

```
sample_goodness_followup_data(
  n = 300,
  origin_year = 1990,
  study_start = as.Date("1990-01-01"),
  study_end = as.Date("2019-12-31"),
  close_date = as.Date("2021-08-06"),
  death_rate = 0.05,
  event_rate = 0.08,
  seed = 42L,
  loss_rate = 0.03
)
```

Arguments

<code>n</code>	Integer number of patients to simulate. Default 300.
<code>origin_year</code>	Integer calendar year corresponding to zero in <code>iv_opyrs</code> . Default 1990.
<code>study_start, study_end</code>	Date (or coercible string) defining the operation date window.
<code>close_date</code>	Date (or coercible string) for the follow-up closing date. Must be $\geq$ <code>study_end</code> .
<code>death_rate</code>	Annual hazard for death (exponential model). Default 0.05 (median survival ~14 years).
<code>event_rate</code>	Annual hazard for the non-fatal event (exponential model). Default 0.08 (median time-to-event ~9 years).
<code>seed</code>	Integer random seed for reproducibility. Default 42.
<code>loss_rate</code>	Annual hazard of loss to active follow-up (exponential model). Default 0.03. 0 means nobody is lost.

Details

Follow-up is deliberately imperfect, so the plot shows what real data look like rather than the ideal. Patients lost to follow-up are censored before `close_date`, the date the dataset was frozen, and fall below the diagonal. No patient has follow-up past `close_date`. Set `loss_rate = 0` for complete follow-up, with every survivor censored at `close_date`.

The column names match the defaults of `hv_followup()`: `iv_opyrs`, `iv_dead`, `dead`. The event-panel columns (`iv_event`, `ev_event`, `deads`) are included so callers can pass `event_col`, `event_time_col`, and `death_for_event_col` directly.

Value

A data frame with columns:

`iv_opyrs` Years from `origin_year` to operation date.

`iv_dead` Follow-up years to death or censoring.

`dead` Logical all-source death indicator.

`iv_event` Follow-up years to non-fatal event or censoring.

ev_event Logical non-fatal event indicator.

deads Logical active/systematic death indicator (subset of dead).

See Also

[hv_followup\(\)](#)

Examples

```
dta <- sample_goodness_followup_data()
head(dta)

# Death panel
gf <- hv_followup(dta)
plot(gf) +
  ggplot2::scale_color_manual(
    values = c("Alive" = "blue", "Dead" = "red"), name = NULL
  ) +
  ggplot2::scale_shape_manual(values = c(1, 4), name = NULL) +
  ggplot2::labs(x = "Operation Date", y = "Follow-up (years)")

# Event panel
gf2 <- hv_followup(dta, event_col = "ev_event",
                  event_time_col = "iv_event",
                  death_for_event_col = "deads")
plot(gf2, type = "event") +
  ggplot2::scale_color_manual(
    values = c("No event" = "blue", "Non-fatal event" = "green3",
              "Death" = "red"),
    name = NULL
  ) +
  ggplot2::scale_shape_manual(values = c(1, 2, 4), name = NULL) +
  ggplot2::labs(x = "Operation Date", y = "Follow-up (years)")
```

sample_hazard_cohort *Sample Subject-Level Cohort Behind the Hazard Examples*

Description

Returns the subject-level survival data that [sample_hazard_empirical\(\)](#) simulates internally and then discards. Use it when a hazard figure needs a numbers-at-risk table: [hv_atrisk\(\)](#) counts subjects still under observation and therefore cannot work from [sample_hazard_data\(\)](#), which is a prediction grid with no subjects in it.

Usage

```
sample_hazard_cohort(
  n = 500,
  time_max = 10,
  groups = NULL,
  shape = 1.5,
  scale = 8,
  seed = 42L
)
```

Arguments

n	Number of simulated subjects per group . Default 500.
time_max	Scale of the follow-up window (years). Default 10. Note that this is not a hard maximum on the returned time: follow-up is administratively truncated at $1.2 * \text{time_max}$, matching the cohort <code>sample_hazard_empirical()</code> fits to, so returned times run up to 12 at the default. It <i>is</i> the upper end of the grid in <code>sample_hazard_data()</code> and of the bins in <code>sample_hazard_empirical()</code> .
groups	NULL for a single group, or a named numeric vector of hazard multipliers matching those passed to <code>sample_hazard_data()</code> . Names must be present, non-empty and distinct; multipliers must be finite and greater than zero.
shape	Weibull shape parameter. Default 1.5.
scale	Weibull scale parameter (years). Default 8.0.
seed	Random seed. Default 42.

Details

For a given `n`, `time_max`, `groups`, `shape`, `scale` and `seed` this draws the *same* cohort that `sample_hazard_empirical()` fits its Kaplan-Meier overlay to, so the counts and the overlay describe the same subjects. Event times are Weibull, censoring is uniform on $[0.2, 1.5] * \text{time_max}$, and follow-up is administratively truncated at $1.2 * \text{time_max}$. When `groups` is supplied each level is an independent draw of `n` subjects, matching `sample_hazard_empirical()`'s balanced-arms convention.

`sample_hazard_data()` is an *analytic* Weibull curve evaluated at the same parameters; it is not fitted to this cohort. The two share a generative model, not an estimation step, so a figure combining them should not be captioned as a model fit to these subjects.

Value

A data frame with one row per subject and columns `time` (follow-up time, years) and `status` (1 event, 0 censored), plus a factor `group` column when `groups` is not NULL.

See Also

`hv_atrisk()`, `hv_atrisk_compose()`, `sample_hazard_empirical()`, `sample_hazard_data()`

Examples

```

coh <- sample_hazard_cohort(n = 500, time_max = 10)
nrow(coh)
mean(coh$status)

# Numbers at risk under a parametric hazard figure. The cohort and the
# curve share n / shape / scale / seed, so the counts belong under it.
arms <- c("No Takedown" = 1.0, "Takedown" = 0.65)
coh2 <- sample_hazard_cohort(n = 400, time_max = 10, groups = arms)
haz <- hv_hazard(
  sample_hazard_data(n = 400, time_max = 10, groups = arms),
  group_col = "group"
)
hv_atrisk_compose(
  plot(haz),
  hv_atrisk(coh2, time = "time", status = "status", group = "group",
    report_times = c(0, 2, 4, 6, 8, 10))
)

```

sample_hazard_data

*Sample Parametric Hazard Model Predictions***Description**

Simulates parametric survival, hazard, and cumulative-hazard predictions on a fine time grid, matching the structure of the SAS predict dataset produced by `tp.hp.dead.sas` and its variants after fitting a Weibull model with `PROC LIFEREG`.

Usage

```

sample_hazard_data(
  n = 500,
  time_max = 10,
  n_points = 500,
  groups = NULL,
  shape = 1.5,
  scale = 8,
  ci_level = 0.95,
  seed = 42L
)

```

Arguments

<code>n</code>	Number of patients used to scale confidence-limit width. Default 500.
<code>time_max</code>	Upper end of the time axis (years). Default 10.
<code>n_points</code>	Number of time points in the prediction grid. Default 500.

groups	NULL for a single curve, or a named numeric vector of hazard multipliers, e.g. <code>c("Control" = 1.0, "Treatment" = 0.7)</code> . A multiplier < 1 means lower hazard (better survival). Analogous to the group indicator in <code>tp.hp.dead.tkdn.stratified.sas</code> . Names must be present, non-empty and distinct; multipliers must be finite and greater than zero.
shape	Weibull shape parameter. $\text{shape} > 1$ gives increasing hazard (late mortality); $\text{shape} < 1$ gives decreasing hazard (early operative mortality). Default 1.5.
scale	Weibull scale parameter (characteristic time in years, i.e. the time at which $S = \exp(-1) \approx 37\%$). Default 8.0.
ci_level	Confidence level for the CI bands. Default 0.95.
seed	Random seed (unused for deterministic Weibull predictions; kept for API consistency with other sample functions). Default 42.

Details

SAS column mapping:

- `time` <- YEARS / iv_dead (prediction grid)
- `survival` <- SSURVIV (predicted survival, 0–100 %)
- `surv_lower` <- SCLLSURV (lower confidence limit on survival)
- `surv_upper` <- SCLUSURV (upper confidence limit on survival)
- `hazard` <- predicted hazard rate (%/year)
- `haz_lower` <- lower confidence limit on hazard
- `haz_upper` <- upper confidence limit on hazard
- `cumhaz` <- cumulative hazard (%; corresponds to $-\log(S) \times 100$)
- `group` <- group stratification variable (when groups is not NULL)

Value

A data frame with columns `time`, `survival`, `surv_lower`, `surv_upper`, `hazard`, `haz_lower`, `haz_upper`, `cumhaz`, and (when groups is not NULL) `group`.

See Also

[hv_hazard\(\)](#), [sample_hazard_empirical\(\)](#), [sample_life_table\(\)](#)

Examples

```
# Single-group predictions (tp.hp.dead.sas)
dat <- sample_hazard_data(n = 500, time_max = 10)
head(dat)

# Two-group predictions (tp.hp.dead.tkdn.stratified.sas)
dat2 <- sample_hazard_data(
  n = 400, time_max = 10,
  groups = c("No Takedown" = 1.0, "Takedown" = 0.65)
)
head(dat2)
```

sample_hazard_empirical

Sample Kaplan-Meier Empirical Points for Hazard Plot Overlay

Description

Simulates patient-level survival data from a Weibull distribution and returns Kaplan-Meier estimates at a small number of binned time points, matching the structure of the SAS plout dataset used as an empirical overlay in `tp.hp.dead.sas` and related templates.

Usage

```
sample_hazard_empirical(
  n = 500,
  time_max = 10,
  n_bins = 6,
  groups = NULL,
  shape = 1.5,
  scale = 8,
  ci_level = 0.95,
  seed = 42L
)
```

Arguments

<code>n</code>	Number of simulated patients. Default 500.
<code>time_max</code>	Upper end of the follow-up window (years). Default 10.
<code>n_bins</code>	Number of time points at which KM is evaluated. Analogous to the discrete annotation points in the SAS templates. Default 6.
<code>groups</code>	NULL for a single group, or a named numeric vector of hazard multipliers matching those passed to <code>sample_hazard_data()</code> . Names must be present, non-empty and distinct; multipliers must be finite and greater than zero.
<code>shape</code>	Weibull shape parameter. Default 1.5.
<code>scale</code>	Weibull scale parameter (years). Default 8.0.
<code>ci_level</code>	Confidence level. Default 0.95.
<code>seed</code>	Random seed. Default 42.

Details

SAS column mapping:

- `time` <- `IV_DEAD / iv_dead` (evaluation time points)
- `estimate` <- `CUM_SURV` (KM survival estimate, 0–100 %)
- `lower` <- `CL_LOWER` (lower 95 % CI)
- `upper` <- `CL_UPPER` (upper 95 % CI)
- `group` <- stratification variable (when `groups` is not NULL)

Value

A data frame with columns time, estimate, lower, upper, and (when groups is not NULL) group.

See Also

[hv_hazard\(\)](#), [sample_hazard_data\(\)](#)

Examples

```
emp <- sample_hazard_empirical(n = 500, time_max = 10, n_bins = 6)
head(emp)

emp2 <- sample_hazard_empirical(
  n = 400, time_max = 10,
  groups = c("No Takedown" = 1.0, "Takedown" = 0.65)
)
head(emp2)
```

sample_life_table	<i>Sample Population Life Table Data</i>
-------------------	--

Description

Generates age-group-specific population survival curves using Gompertz mortality, matching the US population life table overlays used in tp.hp.dead.age_with_population_life_table.sas and tp.hp.dead.uslife.stratified.sas.

Usage

```
sample_life_table(
  age_groups = NULL,
  age_mids = c(55, 72, 85),
  time_max = 10,
  n_points = 100
)
```

Arguments

- age_groups Character vector of age group labels. Default c("<65", "65-80", "\u226580").
- age_mids Numeric vector of representative ages (years) for each group, same length as age_groups. Default c(55, 72, 85).
- time_max Upper end of the time axis (years). Default 10.
- n_points Number of time points. Default 100.

Details**SAS column mapping:**

- time <- prediction time grid (years)
- survival <- SMATCHED (age-group-specific survivorship, 0–100 %)
- group <- age group label (e.g. "<65", "65–80", "\u226580")

Value

A data frame with columns time, survival, and group.

See Also

[hv_hazard\(\)](#), [sample_hazard_data\(\)](#)

Examples

```
# Default: three age groups (<65, 65-80, 80+) using Gompertz mortality
lt <- sample_life_table(time_max = 10)
head(lt)
nlevels(lt$group)    # 3 age groups
range(lt$survival)   # 0-100 % survivorship scale

# Custom strata – two age groups, 15-year follow-up
lt2 <- sample_life_table(
  age_groups = c("Under 70", "70 and over"),
  age_mids   = c(60, 78),
  time_max   = 15
)
levels(lt2$group)
```

sample_longitudinal_counts_data

Sample Longitudinal Counts Data

Description

Builds a pre-aggregated summary data frame of patient and measurement counts at discrete follow-up time windows. The counts are derived by binning the continuous time column from [sample_spaghetti_data\(\)](#), so the two functions share the same underlying simulation.

Usage

```
sample_longitudinal_counts_data(n_patients = 300, max_obs = 6, seed = 42L)
```

Arguments

n_patients	Number of unique patients passed to <code>sample_spaghetti_data()</code> . Default 300.
max_obs	Maximum observations per patient passed to <code>sample_spaghetti_data()</code> . Default 6.
seed	Random seed. Default 42.

Value

A data frame in long format with columns:

- time_label: ordered factor of follow-up windows
- series: "Patients" or "Measurements"
- count: integer count

See Also

`hv_longitudinal()`, `sample_spaghetti_data()`

Examples

```
dta <- sample_longitudinal_counts_data(n_patients = 300, seed = 42)
str(dta)           # time_label (factor), series, count
levels(dta$time_label) # 7 discrete follow-up windows

# Inspect patient counts at each window
subset(dta, series == "Patients")

# Larger cohort
dta2 <- sample_longitudinal_counts_data(n_patients = 1000, seed = 7)
max(dta2$count)      # peak observation count
```

sample_mirror_histogram_data

Generate Sample Data for Mirrored Histogram

Description

Creates a reproducible data frame for testing `hv_mirror_hist()` in either binary-match or weighted IPTW mode. Propensity scores are simulated via a logistic model: control subjects draw their linear predictor from $N(-\text{sep}/2, 1)$ and treated subjects from $N(+\text{sep}/2, 1)$, so the two score distributions overlap in the center while accumulating mass at opposite extremes. Patients at those extremes cannot find a matching partner within the caliper, which naturally reproduces the "many unmatched at the tails" pattern seen in real studies.

Usage

```
sample_mirror_histogram_data(
  n = 500,
  separation = 1.5,
  caliper = 0.05,
  seed = 42L,
  add_weights = FALSE
)
```

Arguments

<code>n</code>	Number of observations per group (default 500).
<code>separation</code>	Numeric; distance between the two group means on the log-odds scale. Larger values push the score distributions further apart and increase the proportion of unmatched patients at the extremes (default 1.5).
<code>caliper</code>	Matching caliper width expressed in propensity-score units (0–1 scale, default 0.05). Treated patients without a control partner within this distance are left unmatched.
<code>seed</code>	Integer random seed for reproducibility (default 42L).
<code>add_weights</code>	Logical. When TRUE an <code>mt_wt</code> column of ATE-style IPTW weights derived from the simulated propensity scores is appended and normalized to mean 1 within each group (default FALSE).

Value

Data frame with columns:

`prob_t` Propensity score on the 0–1 scale.

`tavr` Group indicator (0 = control, 1 = treated).

`match` Binary match indicator produced by greedy nearest-neighbor matching within caliper (1 = matched).

`mt_wt` (Only when `add_weights = TRUE`) ATE IPTW weights normalized to mean 1 within each group.

See Also

[hv_mirror_hist\(\)](#)

Examples

```
# Binary-match mode sample data (default)
dta <- sample_mirror_histogram_data(n = 500, separation = 1.5)
head(dta)
table(dta$tavr, dta$match) # matched vs unmatched counts per group

# IPTW weighted mode – adds mt_wt column
dta_wt <- sample_mirror_histogram_data(n = 500, add_weights = TRUE)
head(dta_wt)
```

```
tapply(dta_wt$mt_wt, dta_wt$tavr, mean) # should be ~1 in each group
```

sample_nnt_data	<i>Sample Number Needed to Treat Data</i>
-----------------	---

Description

Computes the number needed to treat (NNT) and absolute risk reduction (ARR) over time from a two-group survival difference, matching the output of `tp.hp.numtreat.survdiff.matched.sas`.

Usage

```
sample_nnt_data(
  n = 500,
  time_max = 10,
  n_points = 500,
  groups = c(Control = 1, Treatment = 0.7),
  shape = 1.5,
  scale = 8,
  ci_level = 0.95,
  seed = 42L
)
```

Arguments

<code>n</code>	Number of patients per group (used for CI width). Default 500.
<code>time_max</code>	Upper end of the time axis (years). Default 10.
<code>n_points</code>	Number of prediction grid points. Default 500.
<code>groups</code>	Named numeric vector of length 2; hazard multipliers for groups 1 and 2. The group with the smaller multiplier has better survival. Default <code>c("Control" = 1.0, "Treatment" = 0.7)</code> . Names must be present, non-empty and distinct; multipliers must be finite and greater than zero.
<code>shape</code>	Weibull shape. Default 1.5.
<code>scale</code>	Weibull scale (years). Default 8.0.
<code>ci_level</code>	Confidence level. Default 0.95.
<code>seed</code>	Random seed. Default 42.

Details

SAS column mapping:

- `time` <- prediction grid (years)
- `arr` <- absolute risk reduction (survival difference, %)
- `arr_lower` / `arr_upper` <- CI on ARR
- `nnt` <- number needed to treat ($= 100 / \text{ARR}$)
- `nnt_lower` / `nnt_upper` <- CI on NNT (inverted from ARR CI)

Value

A data frame with columns time, arr, arr_lower, arr_upper, nnt, nnt_lower, nnt_upper.

See Also

[hv_nnt\(\)](#), [hv_survival_difference\(\)](#)

Examples

```
nnt_dat <- sample_nnt_data(  
  groups = c("Control" = 1.0, "Treatment" = 0.7)  
)  
head(nnt_dat)
```

sample_nonparametric_curve_data
<i>Sample Nonparametric Curve Data</i>

Description

Simulates pre-computed curve output matching what SAS produces after fitting a two-phase non-parametric temporal trend model and averaging patient-specific profiles with PROC SUMMARY. The output is suitable for direct use with [hv_nonparametric\(\)](#).

Usage

```
sample_nonparametric_curve_data(  
  n = 500,  
  time_max = 12,  
  n_points = 500,  
  groups = NULL,  
  outcome_type = c("probability", "continuous"),  
  ci_level = 0.68,  
  n_bins = 10,  
  seed = 42L  
)
```

Arguments

n	Number of simulated patients (used for binned data points and CI width). Default 500.
time_max	Upper end of the time axis (same units as the SAS iv_echo / iv_wristm variable). Default 12.
n_points	Number of time points on the fine prediction grid. Default 500 (matches the SAS inc=(max-min)/499.9 loop).

groups	NULL for a single average curve, or a named numeric vector of group-specific hazard multipliers, e.g. <code>c("Ozaki" = 0.8, "CE-Pericardial" = 1.2)</code> . Names must be present, non-empty and distinct; multipliers must be finite and greater than zero.
outcome_type	"probability" (binary outcome, 0-1 scale) or "continuous". Default "probability".
ci_level	Confidence level for bootstrap-style CI bands. Default 0.68.
n_bins	Number of equal-sized data-summary bins. Default 10.
seed	Random seed. Default 42.

Details

SAS context: In the SAS templates this dataset corresponds to `mean_curv` (estimate column) plus `boots_ci` (lower/upper columns). Export those datasets to CSV and read them with [read.csv\(\)](#) to use your own model output instead of this sample function.

Value

A data frame with columns `time`, `estimate`, `lower`, `upper`, and (if `groups` is not NULL) `group`.

See Also

[hv_nonparametric\(\)](#), [sample_nonparametric_curve_points\(\)](#)

Examples

```
# Single average curve
dat <- sample_nonparametric_curve_data(n = 500, time_max = 12)
head(dat)

# Two-group comparison
dat2 <- sample_nonparametric_curve_data(
  n = 400, time_max = 7,
  groups = c("Ozaki" = 0.7, "CE-Pericardial" = 1.3),
  outcome_type = "continuous"
)
head(dat2)
```

sample_nonparametric_curve_points

Sample Nonparametric Curve Data Points

Description

Returns only the binned patient-level data summary points from [sample_nonparametric_curve_data\(\)](#). Accepts the same parameters and returns a plain `data.frame`.

Usage

```
sample_nonparametric_curve_points(
  n = 500,
  time_max = 12,
  n_points = 500,
  groups = NULL,
  outcome_type = c("probability", "continuous"),
  ci_level = 0.68,
  n_bins = 10,
  seed = 42L
)
```

Arguments

<code>n</code>	Number of simulated patients (used for binned data points and CI width). Default 500.
<code>time_max</code>	Upper end of the time axis (same units as the SAS <code>iv_echo</code> / <code>iv_wristm</code> variable). Default 12.
<code>n_points</code>	Number of time points on the fine prediction grid. Default 500 (matches the SAS <code>inc=(max-min)/499.9</code> loop).
<code>groups</code>	NULL for a single average curve, or a named numeric vector of group-specific hazard multipliers, e.g. <code>c("Ozaki" = 0.8, "CE-Pericardial" = 1.2)</code> . Names must be present, non-empty and distinct; multipliers must be finite and greater than zero.
<code>outcome_type</code>	"probability" (binary outcome, 0-1 scale) or "continuous". Default "probability".
<code>ci_level</code>	Confidence level for bootstrap-style CI bands. Default 0.68.
<code>n_bins</code>	Number of equal-sized data-summary bins. Default 10.
<code>seed</code>	Random seed. Default 42.

Value

A data frame with columns `time`, `value`, and (if `groups` is not NULL) `group`.

See Also

[sample_nonparametric_curve_data\(\)](#), [hv_nonparametric\(\)](#)

Examples

```
# Single-group data summary points
pts <- sample_nonparametric_curve_points(n = 500, time_max = 12)
head(pts)
names(pts)          # "time", "value"

# Two-group points
pts2 <- sample_nonparametric_curve_points(
  n          = 400,
```

```

time_max      = 7,
groups        = c("Ozaki" = 0.7, "CE-Pericardial" = 1.3),
outcome_type  = "continuous"
)
levels(pts2$group)

```

sample_nonparametric_ordinal_data

Sample Nonparametric Ordinal Curve Data

Description

Simulates pre-computed grade-specific probability curves matching what SAS produces after fitting a cumulative proportional-odds nonparametric temporal trend model. Covers the ordinal TR / AR grade patterns from:

- tp.np.tr.ivecho.average_curv.ordinal.sas (p0, p1, p2, p3 individual probs)
- tp.np.po_ar.u_multi.ordinal.sas (multi-scenario ordinal, p34 = p3+p4)

Usage

```

sample_nonparametric_ordinal_data(
  n = 1000,
  time_max = 5,
  n_points = 500,
  grade_labels = c("Grade 0", "Grade 1", "Grade 2", "Grade 3"),
  n_bins = 10,
  seed = 42L
)

```

Arguments

n	Number of simulated patients (controls CI width and data-point variability). Default 1000.
time_max	Upper end of the time axis (years). Default 5.
n_points	Number of time points on the prediction grid. Default 500.
grade_labels	Character vector of grade labels, one per grade level in ascending order. Default c("Grade 0", "Grade 1", "Grade 2", "Grade 3").
n_bins	Number of equal-sized bins for the data summary points. Default 10.
seed	Random seed. Default 42.

Details

SAS context: The SAS predict dataset has one column per grade (p0, p1, p2, p3) after computing individual probabilities from cumulative probabilities ($cp0 = co0 / (1 + co0)$, etc.). Export it to CSV and reshape to long format to use your own model output.

Value

A long-format data frame: time, estimate, grade (factor). Individual grade probabilities sum to 1 at each time point.

See Also

[hv_ordinal\(\)](#), [sample_nonparametric_curve_data\(\)](#), [sample_nonparametric_ordinal_points\(\)](#)

Examples

```
dat <- sample_nonparametric_ordinal_data(n = 800, time_max = 5)
head(dat)
# verify probabilities sum to 1 at each time point
tapply(dat$estimate[dat$time == dat$time[1]],
       dat$grade[dat$time == dat$time[1]], sum)
```

sample_nonparametric_ordinal_points

Sample Nonparametric Ordinal Data Points

Description

Returns only the binned patient-level data summary points from [sample_nonparametric_ordinal_data\(\)](#). Accepts the same parameters and returns a plain data.frame.

Usage

```
sample_nonparametric_ordinal_points(
  n = 1000,
  time_max = 5,
  n_points = 500,
  grade_labels = c("Grade 0", "Grade 1", "Grade 2", "Grade 3"),
  n_bins = 10,
  seed = 42L
)
```

Arguments

n	Number of simulated patients (controls CI width and data-point variability). Default 1000.
time_max	Upper end of the time axis (years). Default 5.
n_points	Number of time points on the prediction grid. Default 500.
grade_labels	Character vector of grade labels, one per grade level in ascending order. Default c("Grade 0", "Grade 1", "Grade 2", "Grade 3").
n_bins	Number of equal-sized bins for the data summary points. Default 10.
seed	Random seed. Default 42.

Value

A data frame with columns time, value, grade.

See Also

[sample_nonparametric_ordinal_data\(\)](#), [hv_ordinal\(\)](#)

Examples

```
# Default: four grade levels
pts <- sample_nonparametric_ordinal_points(n = 800, time_max = 5)
head(pts)
levels(pts$grade)

# Clinical AR grade labels
pts2 <- sample_nonparametric_ordinal_points(
  n           = 600,
  time_max    = 7,
  grade_labels = c("None", "Mild", "Moderate", "Severe")
)
table(pts2$grade)
```

sample_spaghetti_data *Sample Spaghetti / Profile Plot Data*

Description

Generates a realistic repeated-measures longitudinal data set for demonstrating [hv_spaghetti\(\)](#). Each row is one observation for one patient at one time point, mimicking serial echocardiographic measurements after cardiac surgery (AV mean gradient trajectory over follow-up). Patients have an irregular number of follow-up measurements and a group indicator (e.g. sex), matching the b_echo.xpt structure in the template.

Usage

```
sample_spaghetti_data(
  n_patients = 150,
  max_obs    = 6,
  groups     = c(Female = 0.45, Male = 0.55),
  seed       = 42L
)
```

Arguments

n_patients	Number of unique patients. Default 150.
max_obs	Maximum number of observations per patient. Default 6.
groups	Named character vector of group labels and their sampling probabilities (summing to 1). Default c(Female = 0.45, Male = 0.55).
seed	Random seed for reproducibility. Default 42.

Value

A data frame with columns:

- id: patient identifier (integer)
- time: years from index procedure (numeric)
- value: continuous outcome (numeric; AV mean gradient in mmHg)
- group: group label (factor)

See Also

[hv_spaghetti\(\)](#)

Examples

```
dta <- sample_spaghetti_data(n_patients = 100, seed = 42)
head(dta)
table(dta$group)
```

sample_stacked_histogram_data

Generate Sample Data for Stacked Histogram

Description

Creates a minimal data frame suitable for demonstrating or testing [hv_stacked](#).

Usage

```
sample_stacked_histogram_data(
  n_years = 20,
  start_year = 2000,
  n_categories = 3,
  seed = 42L
)
```

Arguments

n_years	Integer; number of consecutive years to simulate starting from start_year. Defaults to 20.
start_year	Integer; first calendar year in the sequence. Defaults to 2000.
n_categories	Integer; number of distinct groups. Defaults to 3.
seed	Integer passed to set.seed for reproducibility. Defaults to 42.

Value

A data frame with columns year (integer) and category (integer, 1 to n_categories).

See Also[hv_stacked\(\)](#)**Examples**

```
dta <- sample_stacked_histogram_data()
head(dta)
table(dta$year, dta$category)
```

sample_survival_data	<i>Generate Sample Survival Data</i>
----------------------	--------------------------------------

Description

Simulates exponential survival times with administrative censoring at study_years. The default hazard_rate = 0.05 yields roughly 63\

Usage

```
sample_survival_data(
  n = 500,
  hazard_rate = 0.05,
  strata_levels = NULL,
  hazard_ratios = NULL,
  study_years = 20,
  seed = 42
)
```

Arguments

n	Number of observations. Must be a positive integer. Defaults to 500.
hazard_rate	Base annual hazard rate (exponential distribution parameter). Defaults to 0.05.
strata_levels	Optional character vector of stratum labels (e.g. c("Type A", "Type B")). When NULL (the default) a single unstratified cohort is generated.
hazard_ratios	Numeric vector of hazard multipliers, one per element of strata_levels, relative to hazard_rate. Defaults to all 1 (equal hazard across strata). Ignored when strata_levels is NULL.
study_years	Length of administrative follow-up in years. Subjects event-free beyond this point are censored. Defaults to 20.
seed	Integer random seed for reproducibility. Defaults to 42.

Value

A data frame with columns:

`iv_dead` Follow-up time in years (numeric), truncated at `study_years`.

`dead` Logical event indicator; TRUE if the subject experienced the event before administrative censoring.

`iv_opyrs` Operation year offset, uniform over $[1990, 1990 + \text{study_years}]$.

`age_at_op` Age at operation (years); drawn from $N(65, 10)$, capped to $[30, 90]$.

`valve_type` (Only when `strata_levels` is not NULL) Character stratum label.

See Also

Other Kaplan-Meier survival: [hv_survival\(\)](#), [plot.hv_survival\(\)](#)

Examples

```
# Unstratified
dta <- sample_survival_data(n = 500, seed = 42)
head(dta)

# Stratified with differential hazard
dta_s <- sample_survival_data(
  n          = 500,
  strata_levels = c("Type A", "Type B"),
  hazard_ratios = c(1, 1.4),
  seed       = 42
)
table(dta_s$valve_type)
```

sample_survival_difference_data

Sample Survival Difference (Life-Gained) Data

Description

Computes a group-vs-group survival difference curve and confidence interval, matching the output of the HAZDIFL macro used in `tp.hp.dead.life-gained.sas`.

Usage

```
sample_survival_difference_data(
  n = 500,
  time_max = 10,
  n_points = 500,
  groups = c(Control = 1, Treatment = 0.7),
  shape = 1.5,
```

```

    scale = 8,
    ci_level = 0.95,
    seed = 42L
  )

```

Arguments

<code>n</code>	Number of patients per group (used for CI width). Default 500.
<code>time_max</code>	Upper end of the time axis (years). Default 10.
<code>n_points</code>	Number of prediction grid points. Default 500.
<code>groups</code>	Named numeric vector of length 2; hazard multipliers for groups 1 and 2. The group with the smaller multiplier has better survival. Default <code>c("Control" = 1.0, "Treatment" = 0.7)</code> . Names must be present, non-empty and distinct; multipliers must be finite and greater than zero.
<code>shape</code>	Weibull shape. Default 1.5.
<code>scale</code>	Weibull scale (years). Default 8.0.
<code>ci_level</code>	Confidence level. Default 0.95.
<code>seed</code>	Random seed. Default 42.

Details

SAS column mapping:

- `time` <- prediction grid
- `difference` <- `survival(group 2) - survival(group 1)` (percentage points)
- `diff_lower` <- lower CI on difference
- `diff_upper` <- upper CI on difference
- `group1_surv` <- survival curve for group 1
- `group2_surv` <- survival curve for group 2

Value

A data frame with columns `time`, `difference`, `diff_lower`, `diff_upper`, `group1_surv`, `group2_surv`.

See Also

[hv_survival_difference\(\)](#), [sample_hazard_data\(\)](#)

Examples

```

diff_dat <- sample_survival_difference_data(
  groups = c("Control" = 1.0, "Treatment" = 0.7)
)
head(diff_dat)

```

sample_trends_data	<i>Sample Temporal Trend Data</i>
--------------------	-----------------------------------

Description

Generates a realistic patient-level longitudinal data set for demonstrating `hv_trends()`. Each row is one patient with a surgery year, continuous outcome (value), and a grouping variable (group). Trend patterns are modeled so that group means diverge over time, matching the multi-group NYHA / LV-mass / LOS pattern in the SAS template.

Usage

```
sample_trends_data(  
  n = 600,  
  year_range = c(1990L, 2020L),  
  groups = c("Group I", "Group II", "Group III", "Group IV"),  
  seed = 42L  
)
```

Arguments

n	Total number of patients. Default 600.
year_range	Integer vector <code>c(start, end)</code> for the x-axis range (surgery year, or patient age when used with <code>tp.lp.trends.age.sas</code> patterns). Default <code>c(1990, 2020)</code> .
groups	Character vector of group labels, or NULL for a single-group figure (no group column returned; use with <code>hv_trends(..., group_col = NULL)</code>). Default <code>c("Group I", "Group II", "Group III", "Group IV")</code> .
seed	Random seed for reproducibility. Default 42.

Value

- A data frame with columns:
- year: x-axis value (integer; surgery year or patient age)
 - value: continuous outcome (numeric)
 - group: group label (factor, ordered by groups); absent when `groups = NULL`

See Also

[hv_trends\(\)](#)

Examples

```
dta <- sample_trends_data(n = 400, seed = 42)  
head(dta)  
table(dta$year, dta$group)
```

sample_upset_data	<i>Sample Procedure Co-occurrence Data</i>
-------------------	--

Description

Generates a realistic cardiac-surgery procedure data set where each row is a patient and each column is a logical indicator of a specific procedure. Co-occurrence rates are modeled from a latent primary-procedure type so that the UpSet plot shows meaningful overlap patterns (e.g. aortic valve patients frequently have concomitant aorta work; mitral valve patients frequently have concomitant TV repair).

Usage

```
sample_upset_data(n = 500, seed = 42L)
```

Arguments

n	Number of patients. Default 500.
seed	Random seed for reproducibility. Default 42.

Value

A data frame with n rows and the following logical columns: AV_Replacement, AV_Repair, MV_Replacement, MV_Repair, TV_Repair, Aorta, CABG.

See Also

[hv_upset\(\)](#)

Examples

```
dta <- sample_upset_data(n = 300, seed = 42)
head(dta)
colSums(dta)
```

save_manuscript	<i>Save a ggplot at HVTI manuscript defaults</i>
-----------------	--

Description

A thin wrapper around `ggplot2::ggsave()` that enforces the house manuscript figure size, 6 inches wide by 4 inches tall, so journal figures come out at a consistent size in one call. It is the manuscript counterpart of `save_ppt()`, which enforces the slide panel box.

Usage

```

save_manuscript(
  plot,
  file,
  width = 6,
  height = 4,
  units = "in",
  device = NULL,
  dpi = 300,
  draft_file = NULL,
  draft_dpi = NULL,
  format = c("tiff", "png"),
  ...
)

```

Arguments

plot	A <code>ggplot2::ggplot()</code> object, e.g. <code>plot(hv_*)</code> finished with <code>theme_hv_manuscript()</code> .
file	Output file path. Its extension sets the format (<code>.pdf</code> , <code>.tiff</code> , <code>.png</code> , ...). Without an extension, format supplies one.
width, height, units	Figure size. Default 6 x 4 "in", the HVTI manuscript default.
device	Graphics device. NULL (default) lets ggplot2 pick from the file extension, and a TIFF is written LZW-compressed by <code>ragg::agg_tiff()</code> . When you pass a device, file is used as given and format adds no extension. For font embedding in a PDF, pass <code>grDevices::cairo_pdf</code> (requires cairo/X11 support).
dpi	Resolution for raster formats. Default 300.
draft_file	Optional second output file path (typically <code>.png</code>). When supplied, an additional raster copy of plot is written here at the same width/height/units, for dragging into a Word draft manuscript. Default NULL (no draft copy written).
draft_dpi	Resolution for draft_file. Default NULL, which falls back to dpi.
format	Raster format, "tiff" (default) or "png", for a file or draft_file given without an extension. A path that has an extension keeps it. Passing a format that contradicts a raster extension, such as <code>format = "png"</code> with a <code>.tiff</code> path, is an error.
...	Further arguments passed to <code>ggplot2::ggsave()</code> for the primary file. Not applied to draft_file.

Details

Pair it with `theme_hv_manuscript()`, whose 12 pt `base_size` supplies the manuscript typography. Like `save_ppt()`, `save_manuscript()` fixes the output geometry, not the theme; style the plot first, then save.

The device is inferred from the file extension by default. A `.tif` or `.tiff` file is written as an LZW-compressed TIFF at dpi, so a 6 x 4 in figure is 1800 x 1200 pixels, the same geometry as the PNG, without the tens of megabytes an uncompressed TIFF takes. A path with no extension

takes one from `format`, which defaults to `"tiff"`, the raster format journals ask for. An extension you write is never overridden, so a `.png` path, such as one a Quarto or R Markdown report embeds, stays a PNG: browsers do not display TIFF.

For a `.pdf` where you want fonts embedded (so 12 pt type renders exactly as designed), pass `device = grDevices::cairo_pdf` (on a system with `cairo/X11` support).

Publisher-accepted formats, vector (PDF, EPS) or TIFF (a raster format also accepted by journals), often produce large or fragile files when dragged into a Word manuscript. Word has no native PDF-as-picture support and silently converts them to bloated, sometimes unreadable EMF. Pass `draft_file` (typically a `.png` path) to also write a small raster copy alongside file in the same call: keep file as the publisher deliverable submitted to the journal, and drag `draft_file` into the Word draft instead.

Value

Invisibly, the file path, with the extension format added when it had none.

See Also

`save_ppt()` for slides, `theme_hv_manuscript()` for the 12 pt typography, `hv_ggsave_dims()` for fixed-panel sizing. Worked examples live in the HVTI ggplot graphics recipes book, <https://ehrlinger.github.io/hvtiGraphics/>.

Examples

```
p <- plot(hv_survival(sample_survival_data(n = 200, seed = 42))) +
  theme_hv_manuscript()
save_manuscript(p, file.path(tempdir(), "survival.pdf"))

# no extension: a 300 dpi, LZW-compressed survival.tiff
save_manuscript(p, file.path(tempdir(), "survival"))

# the same figure as a PNG
save_manuscript(p, file.path(tempdir(), "survival"), format = "png")

# also write a small draft PNG for dragging into Word
save_manuscript(p, file.path(tempdir(), "survival.pdf"),
  draft_file = file.path(tempdir(), "survival_draft.png"))
```

save_ppt

Save ggplot Objects to an Editable PowerPoint Presentation

Description

Writes one ggplot per slide into a PowerPoint file using `officer::ph_with()` and `rvgr::dml()` so that every plot lands as an editable DrawingML vector graphic: shapes, lines, and text remain selectable in PowerPoint. Plots are placed via `officer::ph_location()` for pixel-exact positioning; titles go into the designated title placeholder via `officer::ph_location_type()`.

Usage

```
save_ppt(
  object,
  template = getOption("hvtiPlotR.ppt_template", system.file("extdata",
    "hv_ppt_template.pptx", package = "hvtiPlotR")),
  powerpoint,
  slide_titles = "Plot",
  layout = "Title and Content",
  master = NULL,
  width = 10.1,
  height = 5.8,
  left = 0,
  top = 1.2,
  panel_box = hv_panel_box(format = .hv_template_format(template)),
  fit = TRUE,
  align = c("panel", "fill")
)
```

Arguments

object	A single ggplot object or a named/unnamed list of ggplot objects. Each element produces one slide. May also be an hv_consort object produced by hv_consort() .
template	Path to an existing .pptx file used as the slide template. Defaults to the hvtiPlotR.ppt_template option when set, otherwise to the template shipped in inst/extdata, whose 16:9 slide fits the default panel_box. See Details.
powerpoint	Output path for the new .pptx file. Required.
slide_titles	A character vector of slide titles. Recycled to the number of plots: supply one string for all slides, or one per plot. Default "Plot".
layout	PowerPoint slide layout name from the template. Default "Title and Content". The bundled templates share six layouts by name; name them by role with hv_ppt_layout() ; "figure", "section", "content" and "two" work here, while "title" and "blank" lack a title placeholder and are for officer::add_slide() . Layout names used through hvtiPlotR 2.8.11 ("Custom Layout", "1_Custom Layout", "1_Title Slide", "1_Title Only") are mapped to their new names, with a one-time message.
master	PowerPoint master name from the template, or NULL to use the template's first available master. Default NULL.
width	Plot width in inches within the slide. Default 10.1. Ignored when panel_box is supplied. Ignored for hv_consort objects, whose dimensions are taken from the object's metadata (set at hv_consort() time).
height	Plot height in inches within the slide. Default 5.8. Ignored when panel_box is supplied. Ignored for hv_consort objects, whose dimensions are taken from the object's metadata (set at hv_consort() time).
left	Distance in inches from the left edge of the slide. Default 0.0. Ignored when panel_box is supplied.

top	Distance in inches from the top of the slide. Default 1.2 (below a standard title bar). Ignored when panel_box is supplied.
panel_box	Named list list(width, height, left, top) describing the panel content area to anchor on every slide (in inches). Each plot is placed by <code>hv_add_plot()</code> , so the panel lands at the same slide coordinates on every slide regardless of axis-label width. Defaults to <code>hv_panel_box()</code> for the template's slide format, read from its slide size: for the 16:9 bundled templates the standard CORR rectangle list(width = 9.98, height = 4.11, left = 2.9, top = 1.85), the same for the dark, light and slate templates; for a 4:3 template the provisional standard rectangle. A template of any other size needs an explicit panel_box. Pass panel_box = NULL to fall back to the fixed width/height/left/ top arguments for every slide (legacy behavior). Ignored for hv_consort objects, which are always placed using their own metadata dimensions.
fit	TRUE (the default) to reduce a plot's text, 2 pt at a time at the PPT themes' 32 pt base size, until its axis text, axis titles, facet strips and plot title lie on the slide and below the slide title; see <code>hv_fit_location()</code> . A message names each reduction, and a warning names a plot that still does not fit at 12 pt. FALSE keeps the plot's own sizes and warns when it runs off the slide. Used only with a panel_box.
align	"panel" (the default) puts every plot's panel on panel_box, so panels line up across slides. "fill" widens each panel so the whole figure spans the slide, centered, with 0.45 in clear on each side; see <code>hv_fit_location()</code> . Used only with a panel_box.

Details

The master HVTI slide template is the blue (dark-room) deck kept on the CORR Study Deliverables SharePoint site, in the Analysis Team library as `Templates/sl.template_hvti_blue.current.pptx`. The package bundles a copy with its example slides removed, refreshed from the master with each release; it is the default and pairs with `theme_hv_ppt_dark()`.

For a light room, a light template is bundled as well; pair it with `theme_hv_ppt_light()`:

```
template <- system.file("extdata", "hv_ppt_template_light.pptx",
                        package = "hvtiPlotR")
```

A slate template (solid #384D60 background, centered white titles) is bundled too; pair it with `theme_hv_ppt_slate()`:

```
template <- system.file("extdata", "hv_ppt_template_slate.pptx",
                        package = "hvtiPlotR")
```

All bundled templates share six layouts by name; see `hv_ppt_layout()`.

To use the live master instead of the bundled copy, point the `hvtiPlotR.ppt_template` option at your OneDrive-synced copy, for example in `.Rprofile`:

```
options(hvtiPlotR.ppt_template = file.path(
  "~/Library/CloudStorage/OneDrive-ClevelandClinic",
```

```

"CORR Study Deliverables - Analysis Team/Templates",
"sl.template_hvti_blue.current.pptx"
))

```

Keep templates out of study folders: a copy made there stops receiving updates.

Value

Invisibly returns the path given by powerpoint.

See Also

Worked recipe with rendered output: <https://ehrlinger.github.io/hvtiGraphics/presentations.html>.

```

hv_add_plot(), rvg::dml(), officer::ph_with(), officer::ph_location(), theme_hv_manuscript()

```

Examples

```

library(ggplot2)

# The bundled copy of the HVTI blue master template (also the default);
# no need to keep a .pptx of your own.
template <- system.file("extdata", "hv_ppt_template.pptx",
                        package = "hvtiPlotR")

# -----
# Recommended workflow: preview with light, save with dark
# -----
# Build the plot with theme_hv_ppt_light() so it renders crisply in the
# RStudio viewer (black text/border on the IDE's white background) while
# you iterate on scales and labels:
p_preview <- ggplot(mtcars, aes(x = wt, y = mpg)) +
  geom_point() +
  labs(x = "Weight", y = "Miles per gallon", title = "Fuel economy") +
  theme_hv_ppt_light()
print(p_preview)          # interactive check

# When the layout looks right, swap to theme_hv_ppt_dark() for the deck
# (white text + black panel against a dark slide template) and save:
p_slide <- p_preview + theme_hv_ppt_dark()
save_ppt(
  object      = p_slide,
  template    = template,
  powerpoint  = file.path(tempdir(), "fuel_economy.pptx"),
  slide_titles = "Fuel Economy by Weight"
)

# List of plots - one slide per plot, individual titles
p2 <- ggplot(mtcars, aes(x = factor(cyl), y = mpg)) +
  geom_boxplot() +
  labs(x = "Cylinders", y = "Miles per gallon") +
  theme_hv_ppt_dark()

```

```

save_ppt(
  object      = list(p_slide, p2),
  template    = template,
  powerpoint  = file.path(tempdir(), "deck.pptx"),
  slide_titles = c("Scatter: fuel economy", "Box: mpg by cylinder count")
)

# No y-axis label – let the slide title carry the meaning. `y = NULL`
# drops the axis title; the numeric tick labels remain.
p_noy <- ggplot(mtcars, aes(x = factor(cyl), y = mpg)) +
  geom_boxplot() +
  labs(x = "Cylinders", y = NULL) +
  theme_hv_ppt_dark()

save_ppt(
  object      = p_noy,
  template    = template,
  powerpoint  = file.path(tempdir(), "mpg_by_cyl.pptx"),
  slide_titles = "Miles per Gallon by Cylinder Count"
)

# Manuscript (white background) for AATS-style presentations
pm <- ggplot(mtcars, aes(x = wt, y = mpg)) +
  geom_point() +
  labs(x = "Weight", y = "Miles per gallon") +
  theme_hv_manuscript()

save_ppt(
  object      = pm,
  template    = template,
  powerpoint  = file.path(tempdir(), "manuscript.pptx"),
  slide_titles = "Fuel Economy"
)

# -----
# panel_box: anchor the plot panel to a fixed rectangle on every slide
# -----
#
# Problem: when plots in a deck have different y-axis ranges
# (e.g. "1.0" vs "4567.2"), axis-label widths differ and shift the
# panel inside a fixed ph_location(). Without a panel_box the plot
# panel drifts between slides – visually jarring against the slide
# background.
#
# Solution: pass `panel_box = list(width = ..., height = ..., left = ...,
# top = ...)` . This describes the PANEL CONTENT AREA itself – the
# rectangle bounded by the axis lines, not the outer plot extent.
# save_ppt() calls [hv_ph_location()] for each plot, measures the
# axis / title / legend chrome around the panel, and adjusts per-slide
# placement so the panel lands at the target rectangle on every slide.
# The plot's OUTER dimensions therefore vary per slide (chrome floats
# around a constant panel), analogous to how [hv_ggsave_dims()] drives

```

```

# ggsave's width/height from a target panel size.
#
# Workflow: build plots with `theme_hv_ppt_light()` so the white panel
# + black border renders crisply in the IDE viewer during development.
# At save time, the PowerPoint template controls the SLIDE background
# (e.g. blue gradient), so a light_ppt plot on a dark slide template
# produces AATS-style white-panel-on-dark-slide decks.

p_small <- ggplot(mtcars, aes(hp, mpg)) +
  geom_point() +
  scale_x_continuous(breaks = seq(0, 400, 100), expand = c(0, 0)) +
  scale_y_continuous(labels = function(x) sprintf("%3.1f", x)) +
  labs(x = "Horsepower", y = "MPG") +
  theme_hv_ppt_light()

p_big <- ggplot(mtcars, aes(hp, mpg)) +
  geom_point() +
  scale_x_continuous(breaks = seq(0, 400, 100), expand = c(0, 0)) +
  scale_y_continuous(labels = function(x) sprintf("%8.1f", x * 10000)) +
  labs(x = "Horsepower", y = "MPG (x10k)") +
  theme_hv_ppt_light()

# Preview each plot interactively (e.g. print(p_small) in RStudio) to
# check decoration before saving.

# Without panel_box: panel drifts between slides because the big-number
# labels eat more horizontal space than the small-number labels.
save_ppt(
  object      = list(p_small, p_big),
  template    = template,
  powerpoint  = file.path(tempdir(), "drifting_deck.pptx"),
  slide_titles = c("Small y-axis", "Big y-axis")
)

# With panel_box: target is an 8.88" x 4.51" panel at slide coordinates
# (2.58", 1.63"). The panel content area lands at exactly that rectangle
# on both slides; axis labels extend outside it as each plot requires.
# The 2.58" left margin leaves room for a wide "99999.9"-style y-axis
# label on dark PPT templates rendered at base_size = 32.
save_ppt(
  object      = list(p_small, p_big),
  template    = template,
  powerpoint  = file.path(tempdir(), "anchored_deck.pptx"),
  slide_titles = c("Small y-axis", "Big y-axis"),
  panel_box   = list(width = 8.88, height = 4.51, left = 2.58, top = 1.63)
)

# Sizing advice: with `fit = TRUE` (the default) a plot whose chrome
# would leave the slide or cross the title has its text reduced until it
# fits; save_ppt() warns only when even 12 pt text does not fit.

```

scale_fill_hv

*Color and fill scales that apply the role rule to each panel***Description**

Discrete scales that color each panel by `hv_role_palette()`: event vermillion, censored blue, missing light gray, and colorblind-safe colors for everything else. Add one to a page of panels with `&` and every panel is colored from its own levels, so a panel without the event still draws its levels from blue.

Usage

```
scale_fill_hv(
  event = NULL,
  censored = NULL,
  missing = "(Missing)",
  na.value = .HV_ROLE_MISSING,
  ...
)

scale_color_hv(
  event = NULL,
  censored = NULL,
  missing = "(Missing)",
  na.value = .HV_ROLE_MISSING,
  ...
)

scale_colour_hv(
  event = NULL,
  censored = NULL,
  missing = "(Missing)",
  na.value = .HV_ROLE_MISSING,
  ...
)
```

Arguments

<code>event, censored</code>	Level names that take the event or the censored color, or NULL for none. Names not in levels are ignored.
<code>missing</code>	Level name that stands for missing values. "(Missing)" is the explicit level <code>hv_eda()</code> adds.
<code>na.value</code>	Color for a real NA. The default is the missing gray, so an NA and a "(Missing)" level look the same.
<code>...</code>	Passed to <code>ggplot2::discrete_scale()</code> : name, labels, guide and so on.

Details

scale_colour_hv() is an alias of scale_color_hv(), the same pairing of British and US names that ggplot2 gives its own color scales.

Colors stay the caller's choice: no constructor or plot() method in this package applies these scales for you.

A panel with more levels than the colorblind-safe palettes hold is drawn in ggplot2's default hue palette instead, with a warning naming its level count, so one variable with many levels does not stop a report. Call hv_role_palette() directly to get an error in that case.

Value

A discrete ggplot2 scale for the color or fill aesthetic.

See Also

hv_role_palette() for the rule itself.

Examples

```
dta <- data.frame(status = c("Alive", "Dead", "Alive", "(Missing)"))
ggplot2::ggplot(dta, ggplot2::aes(status, fill = status)) +
  ggplot2::geom_bar() +
  scale_fill_hv(event = "Dead", censored = "Alive")
```

summary.hv_data

Summarize an hv_data object

Description

Prints the standard one-screen header (via print.hv_data() or the subclass override) and then walks the object's \$tables slot, printing each named auxiliary table with a header. Lets callers see the underlying risk tables, report tables, diagnostics, etc. without having to know the \$tables accessor path.

Usage

```
## S3 method for class 'hv_data'
summary(object, ...)
```

Arguments

object	An hv_data object.
...	Forwarded to the underlying print() method.

Details

Subclasses can override this with a more curated layout, but the default implementation is enough for every shipping hv_data subclass.

summary.hv_data

169

Value

object, invisibly.

Index

- * **Hazard plot**
 - hv_hazard, 40
 - plot.hv_hazard, 98
- * **Kaplan-Meier survival**
 - hv_survival, 73
 - plot.hv_survival, 111
 - sample_survival_data, 155
- * **Nonparametric curves**
 - hv_nonparametric, 49
 - plot.hv_nonparametric, 102
- * **Propensity Score & Matching**
 - hv_mirror_hist, 46
 - hv_rmst_contrast, 63
 - hv_rmst_curves, 65
 - plot.hv_mirror_hist, 100
 - plot.hv_rmst_contrast, 104
 - plot.hv_rmst_curves, 105
 - sample_covariate_balance_data, 134
- * **Spaghetti plot**
 - hv_spaghetti, 70
 - plot.hv_spaghetti, 108
- * **Temporal trends**
 - hv_trends, 76
 - plot.hv_trends, 113
- * **datasets**
 - nonparametric, 86
 - parametric, 87
- * **mirrored histogram propensity score**
 - overlap matching IPTW weighting**
 - mirror_histogram**
 - hv_mirror_hist, 46
- as.data.frame.hv_data, 9
- as.data.frame.hv_data(), 84
- autoplot.hv_data, 10
- autoplot.hv_data(), 84
- base::as.data.frame(), 9
- base::data.frame(), 9
- do.call(), 39, 56, 58
- eda_classify_var, 11, 28, 29
- eda_classify_var(), 7, 30, 31, 136
- eda_select_vars, 12, 29
- eda_select_vars(), 7, 11, 135, 136
- geom_alluvium, 89
- geom_smooth, 93, 113
- geom_stratum, 89
- ggiraph::girafe(), 39
- ggplot, 17, 90, 91, 94, 96, 99, 101–103, 107, 109, 110, 112, 114, 118
- ggplot2::autoplot(), 10
- ggplot2::discrete_scale(), 167
- ggplot2::facet_grid(), 56
- ggplot2::facet_wrap(), 56
- ggplot2::ggplot(), 98, 99, 101, 105, 106, 112, 113, 160
- ggplot2::ggsave(), 38, 39, 56, 58, 159, 160
- ggplot2::scale_color_brewer(), 54
- ggplot2::theme(), 62, 82, 83
- ggplot2::theme_bw(), 80
- grid::gpar(), 85
- hazard_plot(), 8
- hv_add_plot, 13
- hv_add_plot(), 33, 54, 55, 163, 164
- hv_alluvial, 14, 90, 119
- hv_alluvial(), 7, 131
- hv_atrisk, 16, 17, 18
- hv_atrisk(), 138, 139
- hv_atrisk_compose, 16, 17, 17
- hv_atrisk_compose(), 139
- hv_balance, 18, 90, 91, 119
- hv_balance(), 6, 21, 134
- hv_balance_height, 21
- hv_consort, 21
- hv_consort(), 24, 25, 92, 120, 133, 162
- hv_consort_exclude, 23

- hv_consort_exclude(), 22, 25
- hv_consort_patients, 24
- hv_consort_patients(), 26
- hv_consort_start, 25
- hv_consort_start(), 22–25, 120, 133
- hv_consort_summary, 26
- hv_consort_summary(), 25
- hv_correlation_matrix, 27
- hv_eda, 28, 94, 121
- hv_eda(), 7, 11, 12, 30, 31, 67, 135, 136, 167
- hv_eda_pages, 30
- hv_eda_pages(), 95, 121
- hv_fit_location, 32
- hv_fit_location(), 13, 14, 54, 55, 163
- hv_followup, 34, 96, 122
- hv_followup(), 7, 36–38, 136–138
- hv_followup_panels, 36
- hv_followup_panels(), 98, 122
- hv_ggsave_dims, 38
- hv_ggsave_dims(), 6, 56–58, 161
- hv_girafe, 39, 94
- hv_girafe(), 95
- hv_hazard, 40
- hv_hazard(), 7, 89, 98, 99, 123, 141, 143, 144
- hv_legend_inside, 43
- hv_legend_inside(), 62, 63
- hv_longitudinal, 44, 99, 123
- hv_longitudinal(), 7, 145
- hv_mirror_hist, 46, 100, 101, 124
- hv_mirror_hist(), 6, 64, 66, 101, 105, 106, 135, 145, 146
- hv_nnt, 48
- hv_nnt(), 7, 76, 101, 113, 124, 148
- hv_nonparametric, 49, 102
- hv_nonparametric(), 7, 87, 102, 148–150
- hv_ordinal, 52, 103, 125
- hv_ordinal(), 7, 152, 153
- hv_palette, 53
- hv_panel_box, 54
- hv_panel_box(), 13, 14, 32, 33, 59, 163
- hv_panel_size, 56
- hv_ph_location, 57
- hv_ph_location(), 6, 32, 33, 39, 55
- hv_ppt_layout, 58
- hv_ppt_layout(), 6, 162, 163
- hv_ppt_palette, 59
- hv_ppt_palette(), 54, 62, 63, 67, 68
- hv_ppt_series, 61
- hv_ppt_series(), 59, 60
- hv_rmst_contrast, 63
- hv_rmst_contrast(), 47, 66, 101, 104–106, 126, 135
- hv_rmst_curves, 65
- hv_rmst_curves(), 47, 64, 101, 105, 106, 126, 135
- hv_role_palette, 67
- hv_role_palette(), 54, 167, 168
- hv_sankey, 68, 108, 127
- hv_sankey(), 7, 132
- hv_spaghetti, 70, 109, 127
- hv_spaghetti(), 7, 109, 153, 154
- hv_stacked, 71, 110, 128, 154
- hv_stacked(), 6, 155
- hv_survival, 16, 17, 73, 111, 112, 128
- hv_survival(), 6, 112, 156
- hv_survival_difference, 75
- hv_survival_difference(), 7, 48, 101, 113, 148, 157
- hv_theme_dark_ppt (hvtiPlotR-themes), 80
- hv_theme_light_ppt (hvtiPlotR-themes), 80
- hv_theme_manuscript (hvtiPlotR-themes), 80
- hv_theme_poster (hvtiPlotR-themes), 80
- hv_theme_ppt (hvtiPlotR-themes), 80
- hv_trends, 76, 114, 129
- hv_trends(), 7, 114, 158
- hv_upset, 78, 79, 117, 130
- hv_upset(), 7, 159
- hv_venn, 79, 118, 130
- hvtiPlotR (hvtiPlotR-package), 5
- hvtiPlotR-package, 5
- hvtiPlotR-themes, 80
- is_hv_data, 83
- is_hv_data(), 8
- make_footnote, 84
- make_footnote(), 6, 83, 85
- makeFootnote (make_footnote), 84
- makeFootnote(), 6
- nnt_plot(), 8
- nonparametric, 86, 89
- officer::add_slide(), 14, 59, 162
- officer::ph_location(), 33, 57, 58, 161, 164

- `officer::ph_location_type()`, 161
- `officer::ph_with()`, 14, 161, 164
- `officer::read_pptx()`, 13
- `parametric`, 86, 87, 87
- `patchwork::patchworkGrob()`, 58
- `patchwork::plot_annotation()`, 58
- `plot()`, 27
- `plot.hv_alluvial`, 14, 15, 89
- `plot.hv_balance`, 18, 20, 90
- `plot.hv_consort`, 92
- `plot.hv_consort()`, 22
- `plot.hv_correlation_matrix`, 92
- `plot.hv_correlation_matrix()`, 28
- `plot.hv_eda`, 28, 29, 93
- `plot.hv_eda()`, 39, 40, 95
- `plot.hv_eda_pages`, 95
- `plot.hv_eda_pages()`, 30, 31, 40
- `plot.hv_followup`, 34, 36, 96
- `plot.hv_followup()`, 97, 98
- `plot.hv_followup_panels`, 97
- `plot.hv_followup_panels()`, 36, 38
- `plot.hv_hazard`, 98
- `plot.hv_hazard()`, 40–42
- `plot.hv_longitudinal`, 44, 45, 99
- `plot.hv_mirror_hist`, 46, 47, 100
- `plot.hv_mirror_hist()`, 47, 64, 66, 105, 106, 135
- `plot.hv_nnt`, 101
- `plot.hv_nnt()`, 48
- `plot.hv_nonparametric`, 49–51, 102
- `plot.hv_nonparametric()`, 51
- `plot.hv_ordinal`, 52, 53, 103
- `plot.hv_rmst_contrast`, 104
- `plot.hv_rmst_contrast()`, 47, 63, 64, 66, 101, 106, 135
- `plot.hv_rmst_curves`, 105
- `plot.hv_rmst_curves()`, 47, 64–66, 101, 105, 135
- `plot.hv_sankey`, 68, 69, 107
- `plot.hv_spaghetti`, 70, 71, 108
- `plot.hv_spaghetti()`, 71
- `plot.hv_stacked`, 71, 72, 110
- `plot.hv_survival`, 73, 74, 111
- `plot.hv_survival()`, 74, 156
- `plot.hv_survival_difference`, 112
- `plot.hv_survival_difference()`, 75, 76
- `plot.hv_trends`, 76, 77, 113
- `plot.hv_trends()`, 77
- `plot.hv_upset`, 78, 115
- `plot.hv_venn`, 79, 117
- `print.hv_alluvial`, 119
- `print.hv_balance`, 119
- `print.hv_consort`, 120
- `print.hv_consort_tracker`, 120
- `print.hv_data()`, 168
- `print.hv_eda`, 121
- `print.hv_eda_pages`, 121
- `print.hv_followup`, 122
- `print.hv_followup_panels`, 122
- `print.hv_hazard`, 123
- `print.hv_longitudinal`, 123
- `print.hv_mirror_hist`, 124
- `print.hv_nnt`, 124
- `print.hv_nonparametric`, 125
- `print.hv_ordinal`, 125
- `print.hv_rmst_contrast`, 126
- `print.hv_rmst_curves`, 126
- `print.hv_sankey`, 127
- `print.hv_spaghetti`, 127
- `print.hv_stacked`, 128
- `print.hv_survival`, 128
- `print.hv_survival_difference`, 129
- `print.hv_trends`, 129
- `print.hv_upset`, 130
- `print.hv_venn`, 130
- `ragg::agg_tiff()`, 160
- `read.csv()`, 149
- `rvg::dml()`, 13, 161, 164
- `sample_alluvial_data`, 15, 131
- `sample_alluvial_data()`, 9
- `sample_cluster_sankey_data`, 69, 131
- `sample_cluster_sankey_data()`, 9
- `sample_consort_data`, 133
- `sample_correlation_data`, 133
- `sample_correlation_data()`, 28
- `sample_covariate_balance_data`, 20, 134
- `sample_covariate_balance_data()`, 8, 47, 64, 66, 101, 105, 106
- `sample_eda_data`, 29, 135
- `sample_eda_data()`, 9
- `sample_goodness_followup_data`, 34, 36, 136
- `sample_goodness_followup_data()`, 8
- `sample_hazard_cohort`, 138
- `sample_hazard_data`, 140

- `sample_hazard_data()`, 8, 41, 42, 138, 139, 142–144, 157
- `sample_hazard_empirical`, 142
- `sample_hazard_empirical()`, 8, 42, 138, 139, 141
- `sample_life_table`, 143
- `sample_life_table()`, 8, 42, 141
- `sample_longitudinal_counts_data`, 45, 144
- `sample_longitudinal_counts_data()`, 9
- `sample_mirror_histogram_data`, 47, 101, 145
- `sample_mirror_histogram_data()`, 8
- `sample_nnt_data`, 147
- `sample_nnt_data()`, 8, 48
- `sample_nonparametric_curve_data`, 51, 148
- `sample_nonparametric_curve_data()`, 8, 149, 150, 152
- `sample_nonparametric_curve_points`, 149
- `sample_nonparametric_curve_points()`, 8, 149
- `sample_nonparametric_ordinal_data`, 53, 151
- `sample_nonparametric_ordinal_data()`, 8, 152, 153
- `sample_nonparametric_ordinal_points`, 152
- `sample_nonparametric_ordinal_points()`, 8, 152
- `sample_spaghetti_data`, 71, 153
- `sample_spaghetti_data()`, 8, 144, 145
- `sample_stacked_histogram_data`, 72, 154
- `sample_stacked_histogram_data()`, 8
- `sample_survival_data`, 74, 155
- `sample_survival_data()`, 8, 74, 112
- `sample_survival_difference_data`, 156
- `sample_survival_difference_data()`, 8, 76
- `sample_trends_data`, 77, 158
- `sample_trends_data()`, 8
- `sample_upset_data`, 78, 79, 159
- `sample_upset_data()`, 9
- `save_manuscript`, 159
- `save_manuscript()`, 6
- `save_ppt`, 161
- `save_ppt()`, 6, 14, 39, 54, 55, 58, 59, 63, 83, 85, 92, 159–161
- `scale_color_hv` (`scale_fill_hv`), 167
- `scale_color_hv()`, 67, 68
- `scale_colour_hv` (`scale_fill_hv`), 167
- `scale_fill_hv`, 167
- `scale_fill_hv()`, 67, 68
- `scale_x_upset`, 78, 115
- `set.seed`, 154
- `summary.hv_data`, 168
- `summary.hv_data()`, 84
- `survival_difference_plot()`, 8
- `theme`, 44
- `theme_dark_ppt` (`hvtiPlotR-themes`), 80
- `theme_hv_manuscript`, 14, 18, 28, 47, 49, 51, 52, 70, 71, 73, 74, 76, 77, 90, 91, 94, 96, 100–103, 107–110, 112, 114, 117
- `theme_hv_manuscript` (`hvtiPlotR-themes`), 80
- `theme_hv_manuscript()`, 5, 6, 42, 60, 85, 98, 99, 101, 112, 113, 160, 161, 164
- `theme_hv_poster` (`hvtiPlotR-themes`), 80
- `theme_hv_poster()`, 5
- `theme_hv_ppt_dark` (`hvtiPlotR-themes`), 80
- `theme_hv_ppt_dark()`, 6, 55, 60, 61, 63, 163
- `theme_hv_ppt_light` (`hvtiPlotR-themes`), 80
- `theme_hv_ppt_light()`, 6, 55, 60, 61, 63, 82, 163
- `theme_hv_ppt_slate` (`hvtiPlotR-themes`), 80
- `theme_hv_ppt_slate()`, 6, 55, 163
- `theme_light_ppt` (`hvtiPlotR-themes`), 80
- `theme_man` (`hvtiPlotR-themes`), 80
- `theme_manuscript` (`hvtiPlotR-themes`), 80
- `theme_poster` (`hvtiPlotR-themes`), 80
- `theme_ppt` (`hvtiPlotR-themes`), 80